# Relational Lattices: From Databases to Universal Algebra

Tadeusz Litak[a,*], Szabolcs Mikulás[b], Jan Hidders[c]

[a] *Informatik 8, Friedrich-Alexander-Universität Erlangen-Nürnberg Martensstraße 3, 91058 Erlangen, Germany*
[b] *School of Computer Science and Information Systems, Birkbeck, University of London, WC1E 7HX London, UK*
[c] *Delft University of Technology, Elektrotechn., Wisk. and Inform., Mekelweg 4, 2628CD Delft, The Netherlands*

**Abstract**

*Relational lattices* are obtained by interpreting lattice connectives as *natural join* and *inner union* between database relations. Our study of their equational theory reveals that the variety generated by relational lattices has not been discussed in the existing literature. Furthermore, we show that addition of just the *header constant* to the lattice signature leads to undecidability of the quasiequational theory. Nevertheless, we also demonstrate that relational lattices are not as intangible as one may fear: for example, they do form a pseudoelementary class. We also apply the tools of Formal Concept Analysis and investigate the structure of relational lattices via their standard contexts. Furthermore, we show that the addition of typing rules and *singleton constants* allows a direct comparison with *monotonic relational expressions* of Sagiv and Yannakakis.

*Keywords:* relational lattices, relational algebra, database theory, algebraic logic, lattice theory

## 1. Introduction and Motivation

We study a class of lattices with a natural database interpretation proposed by Vadim Tropashko [39, 34, 38]. It does not seem to have attracted the attention of algebraists, even those investigating the connections between algebraic logic and relational databases (see, e.g., Imieliński and Lipski [17] or Duntsch and Mikulás [10]).

The connective *natural join* $\bowtie$ (which we will interpret as lattice meet!) is one of the basic operations of Codd's *(named) relational algebra* [1, 6]. Incidentally, it is also one of its total operations—i.e., defined for all arguments. In general, Codd's "algebra" is only a *partial algebra*: some operations are defined only between relations with suitable headers, e.g., the (set) union or the difference operator. Apart from the issues of mathematical elegance and generality, this partial nature of operations has also unpleasant practical consequences, forcing database systems to perform at least rudimentary type-checking to avoid "crashing" queries [8].

It turns out, however, that it is possible to generalize the union operation to *inner union* $\oplus$ defined on all elements of the algebra and lattice-dual to natural join. This approach appears more natural and has several advantages over the embedding of relational "algebras" in cylindric algebras proposed in [17]. For example, we avoid an artificial uniformization of headers; hence queries formed with the use of proposed connectives enjoy the *domain independence property* [40], [1, Ch. 5]. We discuss d.i.p. and related properties formally in Section 2.1 below.

*Corresponding author
*Email addresses:* `szabolcs@dcs.bbk.ac.uk` (Szabolcs Mikulás), `A.J.H.Hidders@tudelft.nl` (Jan Hidders)
*URL:* `https://www8.informatik.uni-erlangen.de/~litak` (Tadeusz Litak)

### 1.1. Strong Equivalence vs. Algebraic Equivalence

An important difference between notions of equivalence typically studied by algebraists (in particular, in the present paper) and those typically studied by database theorists can be adequately phrased using terminology and apparatus of an early paper by Aho et al. [3, Sec. 2.5]. For algebraists, it is natural to focus on the notion of *algebraic equivalence* [3, Sec. 2.5], which concerns relation variables without fixed relation schemes, i.e., with a variable set of attributes. The notion naturally considered in the database setting, on the other hand, is that of *strong equivalence* [3, Sec. 2.6], where an explicit schema information is given for relation variables. We discuss this further in Section 6 and in particular Section 6.2: the addition of explicit schema information to (an expansion of) our signature allows a direct comparison with *monotonic relational expressions* of Sagiv and Yannakakis [31, Sec. 2.2].

An excellent account relating these two approaches (which, however, is formulated in slightly different terms than Aho et al. [3]) can be found in a work by Van den Bussche and Waller [9]. The paper motivates the study of algebraic equivalences in terms of *logical data independence*. And we believe that a setting where all the basic operations are totally defined (unlike the standard operations of the relational model) is particularly natural and convenient for such a study.

We focus here on the (quasi)equational theory of natural join and inner union. Apart from an obvious mathematical interest, Birkhoff-style equational inference is the basis for certain query optimization techniques where algebraic expressions represent query evaluation plans and are rewritten by the optimizer into equivalent but more efficient expressions. As for *quasiequations*, i.e., definite Horn clauses over equalities, reasoning over many database constraints such as key constraints and inclusion dependencies [1, Ch. 9] (also known as *foreign keys*) can be reduced to quasiequational reasoning. Note that an optimizer can consider more equivalent alternatives for the original expression if it can take the specified database constraints into account.

### 1.2. A Simple Example

To understand the motivation better, consider an example adapted from one of references on *semantic query optimization* [5, Sec. 3.1]. It concerns querying against a view created over tables which are related by integrity constraints. In such a situation, it is often possible to eliminate natural joins, whose computation is always a costly process. Let us see whether this elimination process can be captured somehow in the quasiequational theory we are talking about; this is also a good opportunity to give some intuition regarding our algebraic setup.

Assume our database contains two relations (or *tables*) *supplier* and *nation* which overlap on a single attribute *nationkey*. Crucially, there is an inclusion dependency which in the notation of Abiteboul et al. [1, Ch. 9] would be expressed as

$$\mathit{supplier}\,(\mathit{nationkey}\,) \subseteq \mathit{nation}\,(\mathit{nationkey}\,) \tag{1}$$

This is an expression which can be translated into an equality in our setup. Of course, the claim that inclusion dependencies correspond directly to equations would require some qualification. First, handling *all* inclusion dependencies would require presence of *renaming*; we briefly discuss suitable extensions of our signature in Section 6.4. However, acyclic inclusion dependencies occurring in real life usually are transformed by copying suitable primary keys, which makes renaming redundant.

Second, one has to keep in mind the distinction between strong equivalence and algebraic equivalence discussed above. We have assumed that headers of *supplier* and *nation* overlap precisely on the attribute(s) where the inclusion dependency holds. Such a situation, irrespective of concrete names of these overlapping attributes, is an instance of

$$s \oplus n = (\mathsf{H} \ltimes s) \oplus n. \tag{2}$$

In this particular case, the instance in question is (in the notation of Section 6 and Table 2)

$$\mathbf{supplier} \oplus \mathbf{nation} = (\mathsf{H} \ltimes \mathbf{supplier}\,) \oplus \mathbf{nation}\,. \tag{3}$$

Concrete databases and relations therein, though, come with an explicit scheme—and inclusion dependency in general does not have to use all the overlapping attributes. This can be handled by adding *unary selection constants* proposed in Section 6.1 and Section 6.2 discusses the rôle of scheme information in this setting. The corresponding equation would be then (again, in the notation of Section 6 and Table 2)

$$\mathbf{supplier} \oplus \mathbf{nation} \oplus \underline{\mathsf{nationkey}} = \mathbf{nation} \oplus \underline{\mathsf{nationkey}}\,. \tag{4}$$

Returning to our example database, assume now we create a view over it as:

**create view** *SupplierInfo*(*name*,*address*,*country*) **as**
**select** *supplier.name*, *supplier.address*, *nation.nname*
**from** *supplier*, *nation*
**where** *supplier.nationkey* = *nation.nationkey*

Views, just like queries, correspond to *terms* in the algebraic setting. In this case, assuming again just like when rendering our inclusion dependency 1 as Equation 3 that *nationkey* is *precisely* where the two tables overlap, the term in question is defined as:

$$\mathbf{SupplierInfo} := (\mathbf{supplier} \bowtie \mathbf{nation}) \oplus (\underline{\mathsf{name}} \bowtie \underline{\mathsf{address}} \bowtie \underline{\mathsf{nname}}).$$

If we assume instead that there may be other attributes where *supplier* and *nation* overlap (i.e., when we choose to render 1 as 4), we would need to resort to one of ways of handling equality-based selection queries proposed in Sections 6.3 and 6.4. Henceforth, we dispose with this additional level of generality.

Finally, assume one formulates the following query:

**select** *name*, *address* **from** *SupplierInfo*

The query corresponds to the term

$$t_1 := \mathbf{SupplierInfo} \oplus (\underline{\mathsf{name}} \bowtie \underline{\mathsf{address}}).$$

The basic equational theory of lattices immediately allows deducing

$$t_1 = (\mathbf{supplier} \bowtie \mathbf{nation}) \oplus (\underline{\mathsf{name}} \bowtie \underline{\mathsf{address}}),$$

which is already a small example of using equational reasoning in optimizing query plans. However, as discussed by Cheng et al. [5, Sec. 3.1], a naïve strategy strategy of query evaluation which would lead to computing **SupplierInfo** first before its substitution into $t_1$ is suboptimal in presence of Equation 3. On the SQL side, one notes that in presence of dependency 1, the above query can be reformulated as

**select** *name*, *address* **from** *Supplier*

avoiding join evaluation altogether. On the algebraic side, the correctness of this rewrite corresponds to validity of quasiequation

$$(3) \;\Rightarrow\; t_1 = \mathbf{supplier} \oplus (\underline{\mathsf{name}} \bowtie \underline{\mathsf{address}}),$$

i.e., to spell out details

$$\begin{aligned} &\mathbf{supplier} \oplus \mathbf{nation} = (\mathsf{H} \bowtie \mathbf{supplier}) \oplus \mathbf{nation} \;\Rightarrow \\ &\qquad (\mathbf{supplier} \bowtie \mathbf{nation}) \oplus (\underline{\mathsf{name}} \bowtie \underline{\mathsf{address}} \bowtie \underline{\mathsf{nname}}) \oplus (\underline{\mathsf{name}} \bowtie \underline{\mathsf{address}}) = \mathbf{supplier} \oplus (\underline{\mathsf{name}} \bowtie \underline{\mathsf{address}}) \end{aligned}$$

and generalizing back from Equation 3 to Equation 2 this is in an instance of

$$s \oplus n = (\mathsf{H} \bowtie s) \oplus n \;\Rightarrow\; s \bowtie n \oplus \underline{\mathsf{a_1}} \bowtie \ldots \bowtie \underline{\mathsf{a_{n+k}}} \oplus \underline{\mathsf{a_1}} \bowtie \ldots \bowtie \underline{\mathsf{a_n}} = s \oplus \underline{\mathsf{a_1}} \bowtie \ldots \bowtie \underline{\mathsf{a_n}}.$$

The question of decidability of quasiequational theory is then a question of the existence of general algorithm to decide validity of such implicational statements.

### *1.3. Initial Hopes*

There were some indications that the considered choice of connectives may lead to positive results concerning decidability/axiomatizability, even for quasiequational theories. On the database side, expressions of our formalisms are closely related to *(unions) of conjunctive queries* [1, Ch. 4], [4] and even more so to *monotonic relational expressions* of Sagiv and Yannakakis [31]; the relationship with these classes will be discussed in more detail in Section 6 below. Such classes of queries admit decision procedures for problems of containment and equivalence based on so-called Homomorphism Theorem [4, 31], [1, Ch. 6]. In fact, Johnson and Klug [21] show that even in presence of *inclusion dependencies*, the containment problem for conjunctive queries remains in NP when infinite database instances are allowed—and presence of inclusion dependencies gives the containment problem distinctly quasi-equational character.

Another reason for our initial optimism came from algebraic logic itself: a somewhat (unjustly!) forgotten book of Craig [7] showed that the *finitization problem* of algebraic logic allows a positive solution when relations are allowed to contain tuples of varying arity. Note that Craig's setting was even more liberal than our present one: while we do happily allow relations with differing headers, we assume that all tuples within one relation are defined on a fixed set of attributes.

### *1.4. Our Findings*

To our surprise, it turned out that relational lattices do not seem to fit anywhere into the rather well-investigated landscape of equational theories of lattices [19, 20]; we will discuss this in detail in Section 3 below. This was in fact what triggered our intensive interest in pursuing this line of work.

To our still greater surprise, it turned out that — at least when it comes to decidability — expansions of relational lattices share the curse of "untamed" structures from algebraic logic such as Tarski's relation algebras or cylindric algebras. As soon as an additional *header constant* $\mathsf{H}$ is added to the language, one can encode the word problem for semigroups in the quasiequational theory using a technique introduced by Maddux [25]. This means that decidability of query equivalence under constraints for restricted positive database languages does not translate into decidability of corresponding quasiequational theories. However, our Theorem 4.7 and Corollary 4.8 do not rule out possible finite axiomatization results (except for quasiequational theory of *finite* structures) or decidability of the equational theory.[1] And with $\mathsf{H}$ removed, i.e., in the pure lattice signature, the picture is completely open. Of course, such a language would be rather weak from a database point of view, but natural for an algebraist.

We also obtain a number of positive results. First of all, concrete relational lattices are pseudoelementary and hence their closure under subalgebras and products is a quasivariety—Theorem 4.1 and Corollary 4.3. The proof yields an encoding into a sufficiently rich (many-sorted) first-order theory with finitely many axioms. This opens up the possibility of using generic proof assistants like Isabelle or Coq in future investigations—so far, we have only used Prover9/Mace4 to study interderivability of interesting (quasi)equations.[2] We have also used the tools of Formal Concept Analysis (Theorem 5.3) to investigate the structure of full concrete relational lattices and establish, e.g., their subdirect irreducibility (Corollary 5.4). Theorem 5.3 is likely to have further applications—see the discussion of Problem 7.1.

This work is a significantly extended and rewritten version of a RAMiCS 2014 paper [24]. The structure of the paper is as follows. In Section 2.1, we provide basic definitions, including the notion of *domain independence* and its natural strengthening *strict independence* (which does not seem to have been explicitly defined before). In Section 2.2, we establish that relational lattices are indeed lattices and in Section 2.3, we note in passing a potential connection with category theory. Section 3 reports our findings about the equational theory of relational lattices: the failure of most standard properties such as weakening of distributivity (Theorem 3.2), those surprising equations and properties that still hold (Theorem 3.5) and dependencies between them (Theorem 3.4). In Section 4, we focus on quasiequations and prove some of most

[1]Note, however, that an extension of our signature to a language with EDPC or a discriminator term would result in an undecidable *equational* theory.

[2]It is worth mentioning that the database inventor of relational lattices has in the meantime developed a dedicated tool [39].

interesting results discussed above, both positive (Theorem 4.1 and Corollaries 4.2–4.4) and negative ones (Theorem 4.7 and Corollaries 4.8–4.9). Section 5 analyzes *standard contexts*, *incidence* and *arrow relations* [12] of relational lattices. Section 6 discusses possible extensions of the signature leading towards *expressive completeness* and addition of typing information, which in turn allows a direct comparison with the setting of (monotonic) relational expressions. Section 7 concludes and discusses future work.

## 2. Basic Definitions

### *2.1. Domains, Relations and Independence*

Let $\mathcal{A}$ be a set of *attribute names* and $\mathcal{D}$ be a set of *domain values.* For $H \subseteq \mathcal{A}$, a *$H$-sequence from $\mathcal{D}$* or an *$H$-tuple over $\mathcal{D}$* is a function $x : H \to \mathcal{D}$, i.e., an element of $\mathcal{D}^H$. $H$ is called the *header* of $x$ and denoted as $h(x)$. The *restriction of $x$ to $H'$* is defined as $x[H'] := \{(a, v) \in x \mid a \in H'\}$, in particular $x[H'] = \emptyset$ if $H' \cap h(x) = \emptyset$. We generalize this to the *projection of a set of $H$-sequences $X$ to a header $H'$* which is $X[H'] := \{x[H'] \mid x \in X\}$. In several places, we will use notation $\{(a_1 : d_1), \ldots, (a_n : d_n)\}$ for an element $x \in \mathcal{D}^{\{a_1,\ldots,a_n\}}$ s.t. $x(a_i) = d_i$.

A *relation* is a pair $r = (H_r, B_r)$, where $H_r \subseteq \mathcal{A}$ is the *header* of $r$ and $B_r \subseteq \mathcal{D}^{H_r}$ is the *body* of $r$. The collection of all relations over $\mathcal{D}$ whose headers are contained in $\mathcal{A}$ will be denoted as $R(\mathcal{D}, \mathcal{A})$. Define the proper class $\mathcal{F} := \{R(\mathcal{D}, \mathcal{A}) \mid \mathcal{D}, \mathcal{A} \in \mathsf{Set}\}$. For a fixed $\mathcal{A} \in \mathsf{Set}$, it is also convenient to isolate the subclass of $\mathcal{F}$ determined by it, i.e., $\mathcal{F}_\mathcal{A} := \{R(\mathcal{D}, \mathcal{A}) \mid \mathcal{D} \in \mathsf{Set}\}$; we have thus $\mathcal{F} = \bigcup_{\mathcal{A}\in\mathsf{Set}} \mathcal{F}_\mathcal{A}$. A *(n-ary) relational query* is a $n$-ary operation $\phi$ defined on all members of $\mathcal{F}$:

$$R(\mathcal{D}, \mathcal{A})^n \ni (r_1, \ldots, r_n) \mapsto \phi^{\mathcal{D},\mathcal{A}}(r_1, \ldots, r_n) \in R(\mathcal{D}, \mathcal{A}).$$

We say that a query $\phi$ is *domain independent* [40], [1, Ch. 5] if for all $\mathcal{D}, \mathcal{D}', \mathcal{A}$, it holds that $\phi^{\mathcal{D},\mathcal{A}}(r_1, \ldots, r_n) = \phi^{\mathcal{D}',\mathcal{A}}(r_1, \ldots, r_n)$ whenever $r_i \in R(\mathcal{D}_i, \mathcal{A}) \cap R(\mathcal{D}'_i, \mathcal{A})$ $(i \in \{1, \ldots, n\})$.

For the purpose of the discussion in Section 6, it is also convenient to define explicitly a stronger property, which appears to be taken for granted in references like [1, Ch. 5]. Namely, say that a query $\phi$ is *strictly independent* if for all $\mathcal{D}, \mathcal{D}', \mathcal{A}, \mathcal{A}'$, it holds that $\phi^{\mathcal{D},\mathcal{A}}(r_1, \ldots, r_n) = \phi^{\mathcal{D}',\mathcal{A}'}(r_1, \ldots, r_n)$ whenever $r_i \in R(\mathcal{D}_i, \mathcal{A}) \cap R(\mathcal{D}'_i, \mathcal{A}')$ $(i \in \{1, \ldots, n\})$. That is, the outcome of $\phi$ is not only independent of irrelevant domain elements, but also of irrelevant attributes. In most of the paper, the operations under consideration are strictly independent (Lemma 2.2 below); only in Section 6.3 we will see an example of domain independent operation which is not strictly independent. We believe this notion is an appropriate strengthening of d.i.p. in the context of *logical data independence* [9] and *algebraic equivalences* [3] (recall the corresponding discussion in Section 1.1).

Examples of queries which do not have even the weaker property of domain independence abound in any references discussing explicitly the difference between first-order calculus and relational algebra (which is domain-independent by design), see Abiteboul et al. [1, Ch. 5] for references. Typical examples involve unrestricted negation or universal quantification. This is not a trivial property from the point of view of first-order logic: Vardi [40] shows that for first-order queries, the property of being domain-independent is undecidable.

### *2.2. Introducing Relational Lattices*

For the relations $r, s$, we define the *natural join* $r \bowtie s$, and *inner union* $r \oplus s$:

$$\begin{aligned} r \bowtie s &:= (H_r \cup H_s, \{x \in \mathcal{D}^{H_r \cup H_s} \mid x[H_r] \in B_r \text{ and } x[H_s] \in B_s\}) \\ r \oplus s &:= (H_r \cap H_s, \{x \in \mathcal{D}^{H_r \cap H_s} \mid x \in B_r[H_s] \text{ or } x \in B_s[H_r]\}) \end{aligned}$$

The definition of natural join naturally suggests a partial operation of *tuple concatenation*: given $x \in \mathcal{D}^{H_r}$ and $y \in \mathcal{D}^{H_s}$, their concatenation is defined if they coincide on $H_r \cap H_s$ and is then the uniquely determined tuple from $\mathcal{D}^{H_r \cup H_s}$ which projects to $x$ and $y$. We could even reuse the natural join symbol for tuple concatenation, but in what follows, there is hardly any need for this.

$$\begin{array}{|cc|}\hline \mathsf{a} & \mathsf{b} \\ \hline 1 & 1 \\ 2 & 2 \\ 3 & 2 \\ 3 & 3 \\ \hline \end{array} \bowtie \begin{array}{|cc|}\hline \mathsf{b} & \mathsf{c} \\ \hline 1 & 1 \\ 2 & 2 \\ 2 & 3 \\ 4 & 4 \\ \hline \end{array} = \begin{array}{|ccc|}\hline \mathsf{a} & \mathsf{b} & \mathsf{c} \\ \hline 1 & 1 & 1 \\ 2 & 2 & 2 \\ 2 & 2 & 3 \\ 3 & 2 & 2 \\ 3 & 2 & 3 \\ \hline \end{array} \qquad \begin{array}{|cc|}\hline \mathsf{a} & \mathsf{b} \\ \hline 1 & 1 \\ 2 & 2 \\ 3 & 2 \\ 3 & 3 \\ \hline \end{array} \oplus \begin{array}{|cc|}\hline \mathsf{b} & \mathsf{c} \\ \hline 1 & 1 \\ 2 & 2 \\ 2 & 3 \\ 4 & 4 \\ \hline \end{array} = \begin{array}{|c|}\hline \mathsf{b} \\ \hline 1 \\ 2 \\ 3 \\ 4 \\ \hline \end{array}$$

Figure 1: Natural join and inner union. In this example, $\mathcal{A} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}\}, D = \{1, 2, 3, 4\}$.

In our notation, $\bowtie$ always binds stronger than $\oplus$. The *header constant* $\mathsf{H} := (\emptyset, \emptyset)$ plays a special rôle: for any $r$, $(H_r, B_r) \bowtie \mathsf{H} = (H_r, \emptyset)$ and hence $r_1$ and $r_2$ have the same headers iff $\mathsf{H} \bowtie r_1 = \mathsf{H} \bowtie r_2$. Note also that the projection of $r_1$ to $H_{r_2}$ can be defined as $r_1 \oplus (\mathsf{H} \bowtie r_2)$. In fact, we can identify $\mathsf{H} \bowtie r$ and $H_r$. We denote $(R(\mathcal{D}, \mathcal{A}), \bowtie, \oplus, \mathsf{H})$ as $\mathfrak{R}^{\mathsf{H}}(\mathcal{D}, \mathcal{A})$, with $\mathcal{L}_{\mathsf{H}} := \{\bowtie, \oplus, \mathsf{H}\}$ denoting the corresponding algebraic signature. $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ is its reduct to the signature $\mathcal{L} := \{\bowtie, \oplus\}$.

**Lemma 2.1.** *For any $\mathcal{D}$ and $\mathcal{A}$, $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ is a lattice.*

*Proof.* This result is due to Tropashko [39, 34, 38], but let us provide an alternative proof. Define $\mathit{Dom} := \mathcal{A} \cup \mathcal{D}^{\mathcal{A}}$ and for any $X \subseteq \mathit{Dom}$ set

$$Cl(X) := X \cup \{x \in \mathcal{D}^{\mathcal{A}} \mid \exists y \in (X \cap \mathcal{D}^{\mathcal{A}}).\, x[\mathcal{A} - X] = y[\mathcal{A} - X]\}.$$

In other words, $Cl(X)$ is the sum of $X \cap \mathcal{A}$ (the set of attributes contained in $X$) with the cylindrification of $X \cap \mathcal{D}^{\mathcal{A}}$ along the axes in $X \cap \mathcal{A}$. It is straightforward to verify $Cl$ is a closure operator and hence $Cl$-closed sets form a lattice, with the order being obviously $\subseteq$ inherited from the powerset of $\mathit{Dom}$. It remains to observe $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ is isomorphic to this lattice: the isomorphism is given by

$$(H, B) \mapsto (\mathcal{A} - H) \cup \{x \in \mathcal{D}^{\mathcal{A}} \mid x[H] \in B\}.$$

$\square$

We call $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ *the (full) relational lattice over ($\mathcal{D}$,$\mathcal{A}$).* We also use the alternative name *Tropashko lattices* to honor the inventor of these structures. The lattice order given by $\bowtie$ and $\oplus$ is

$$(H_r, B_r) \sqsubseteq (H_s, B_s) \text{ iff } H_s \subseteq H_r \text{ and } B_r[H_s] \subseteq B_s.$$

For classes of algebras, we use $\mathbb{H}, \mathbb{S}, \mathbb{P}$ to denote closures under, respectively, homomorphisms, (isomorphic copies of) subalgebras and products. Let

$$\mathcal{R}^{\mathsf{H}}_{\text{fin}} := \mathbb{S}\{\mathfrak{R}^{\mathsf{H}}(\mathcal{D}, \mathcal{A}) \mid \mathcal{D}, \mathcal{A} \text{ finite}\},\ \mathcal{R}^{\mathsf{H}}_{\text{unr}} := \mathbb{S}\{\mathfrak{R}^{\mathsf{H}}(\mathcal{D}, \mathcal{A}) \mid \mathcal{D}, \mathcal{A} \text{ unrestricted}\}$$

and let $\mathcal{R}_{\text{fin}}$ and $\mathcal{R}_{\text{unr}}$ denote the lattice reducts of the respective classes.

**Lemma 2.2.** *All the operations in the signature $\mathcal{L}_{\mathsf{H}}$ are strictly independent.*

*Proof.* Straightforward. $\square$

*2.3. Relational Lattices, (Op-)Fibrations and the Grothendieck Construction*

Let us make a simple aside observation, which seems of independent interest but is not directly relevant for proofs of results in this paper. Given $\mathcal{D}$ and $\mathcal{A}$, a category theorist may note that $H_{(\cdot)}$, i.e., the mapping sending every relation $r = (H_r, B_r)$ to its header $H_r$ is a *(Grothendieck) opfibration* from $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ ordered by $\sqsubseteq$ to $\mathcal{P}^{\supseteq}(\mathcal{A})$, the latter being of course the poset with reverse inclusion order. As we are talking about posets here, the action of $H_{(\cdot)}$ on arrows and its functoriality are obvious. However, as most standard references

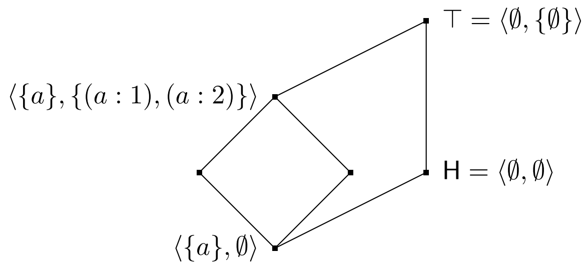


Figure 2: The lattice $\mathfrak{R}(\{0,1\},\{a\})$, i.e., $L_4$ [26, 20]

in category theory (see in particular Jacobs [18, Ch. 1]) introduce opfibrations well after fibrations, it is easier to pattern-match all results and notions without having to reverse arrows all the time. So we recall that opfibration $E \to D$ is just a fibration $E^{op} \to D^{op}$ and therefore our observation can be reformulated as $H_{(\cdot)}$ being a *(Grothendieck) fibration* of $\mathfrak{R}^{\sqsupseteq}(\mathcal{D},\mathcal{A})$ over $\mathcal{P}^{\subseteq}(\mathcal{A})$, where $\mathfrak{R}^{\sqsupseteq}(\mathcal{D},\mathcal{A})$ is $\mathfrak{R}(\mathcal{D},\mathcal{A})$, but ordered by $\sqsupseteq$ rather than $\sqsubseteq$. The crucial thing to note is that an arrow $r' \sqsupseteq r$ is "cartesian" [18, Def. 1.1.3] iff $B_{r'} = B_r[H_{r'}]$, i.e., if $r'$ is the projection of $r$ to the header of $r'$. With this, one can note that $H_{(\cdot)}$ is in fact a *split fibration* [18, Def. 1.4.3].

It is thus most natural to view relational lattices themselves as obtained via the so-called Grothendieck construction [18, Def. 1.10] associated with this particular fibration. This construction is obtained via the *quasifunctor* or *pseudo-functor* [18, Def. 1.4.4] defined as follows

$$\begin{aligned} F^{\mathcal{A}}_{\mathcal{D}} :\ & \mathcal{P}^{\supseteq}(\mathcal{A}) \ni H \longrightarrow \mathcal{P}^{\supseteq}(\mathcal{D}^H) \in \mathsf{Pos} \\ F^{\mathcal{A}}_{\mathcal{D}}(H \supseteq H') :=\ & (\mathcal{D}^H \supseteq B \mapsto B[H'] \subseteq \mathcal{D}^{H'}). \end{aligned}$$

We get then that $\mathfrak{R}^{\sqsupseteq}(\mathcal{D},\mathcal{A})$ is the Grothendieck completion $\int_{\mathcal{P}^{\subseteq}(\mathcal{A})} F^{\mathcal{A}}_{\mathcal{D}}$.

Curiously enough, a number of recent references mentioned (op-)fibrations and the Grothendieck construction in the database context [23, 22, 35]. The focus and the use seems somehow different: that connection arose in the study of queries, views and RDF triples, but it would be interesting to connect it with the Grothendieck perspective on relational lattices sketched above. Our personal belief is that there is even a closer relationship with the categorical approach to relational databases proposed recently by Abramsky [2], which moreover yields a surprising connection with Bell's Theorem from theoretical physics. This belief is motivated by the central rôle played by $\bowtie$ in Abramsky's work [2, Sec. 2.2] and other similarities. It is worth noting that Abramsky [2, Sec. 3] suggests that this categorical approach may yield a natural connection with (and unifying perspective on) probabilistic databases and provenance semirings.

## 3. Towards the Equational Theory of Relational Lattices

Let us begin the section with

**Open Problem 3.1.** *Are* $\mathbb{SP}(\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}) = \mathbb{HSP}(\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}})$ *and* $\mathbb{SP}(\mathcal{R}_{\mathrm{unr}}) = \mathbb{HSP}(\mathcal{R}_{\mathrm{unr}})$*?*

If the answer is "no", it would mean that relational lattices should be considered a quasiequational rather than equational class (cf. Corollary 4.3 below). Note also that the decidability of equational theories seems of less importance from a database point of view than decidability of quasiequational theories. Nevertheless, relating to already investigated varieties of lattices seems a good first step. It turns out that weak forms of distributivity and similar properties studied in standard references [19, 20, 37] tend to fail dramatically:

**Theorem 3.2.** $\mathcal{R}_{\text{fin}}$ *(and hence* $\mathcal{R}_{\text{unr}}$*) does not have any of the following properties (see the above references or the proof below for definitions):*

1. *upper- and lower-semidistributivity,*
2. *almost distributivity and neardistributivity,*
3. *upper- or lower-semimodularity (and hence also modularity),*
4. *the Jordan–Dedekind chain condition,*
5. *supersolvability.*

*Proof.* For most clauses, it is enough to observe that $\mathfrak{R}(\{0,1\},\{0\}))$ is isomorphic to $L_4$, a lattice generating a variety covering the non-modular variety generated by $N_5$ [26, 20]: a routine counterexample in such cases, see Figure 2.

In more detail:

Clause 1: Recall that *join-semidistributivity* is the property:

$a \oplus b = a \oplus c$ implies $a \oplus b = a \oplus (b \Join c)$.

Now take $a$ to be $\mathsf{H}$ and $b$ and $c$ to be atoms with the header $\{0\}$.

Clause 2: This is a corollary of Clause 1 [19, Th 4.2 and Sec 4.3].

Clause 3: Recall that *semimodularity* is the property:

if $a \Join b$ is covered $a$ and $b$, then $a \oplus b$ covers $a$ and $b$.

Again, take $a$ to be $\mathsf{H}$ and $b$ to be either of the atoms with the header $\{0\}$.

Clause 4: Recall that *the Jordan-Dedekind chain condition* is the property that the cardinalities of two maximal chains between common end points are equal. This obviously fails in $N_5$ and $L_4$.

Clause 5: Recall that for finite lattices, *supersolvability* [36] boils down to the existence of a maximal chain generating a distributive lattice with any other chain. Again, this fails in $N_5$ and $L_4$. $\square$

**Remark 3.3.** *Theorem 3.2 has an additional consequence regarding the notion called rather misleadingly* boundedness *in most standard references (see e.g., Jipsen and Rose [19, p. 27]): being an image of a freely generated lattice by a* bounded morphism. *We use the term* McKenzie-bounded*, as McKenzie showed that for finite subdirectly irreducible lattices, this property amounts to splitting the lattice of varieties of lattices [19, Theorem 2.25]. Finite Tropashko lattices are subdirectly irreducible (Corollary 5.4 below) but Clause 1 of Theorem 3.2 entails they are not McKenzie-bounded [19, Lemma 2.30].*

Nevertheless, Tropashko lattices do not generate the variety of all lattices. The results of our investigations so far on valid (quasi)equations are summarized in the remainder of this section.

Table 1 presents a number of equations and quasiequations that hold in Tropashko lattices. Theorem 3.5 below states formally their validity, but before we prove it, let us say a few words about motivations for these sentences and their mutual dependencies.

It is instructive to consider the axiom(s) of distributive lattices. In our relational setting, the distributive law holds only in special situations. Clearly, it is valid when all the elements involved have the same header, but in fact one can weaken this assumption considerably. Quasiequations Qu1, Qu2 and Qu3 show how far it can be weakened. The first of these, i.e., Qu1 is a form of weak distributivity, denoted as $CD_\vee$ [28] or $WD_\wedge$ [20]. Of course, when we consider just the interactions between headers of elements and completely ignore bodies of relations, distributivity laws hold: these are laws Eq1, Eq3 and Eq2.

In order to see what remains of distributivity in the general case and how much can be expressed by means of equations (as opposed to quasiequations), compare AxRH2 and RL1. The two sides of the distributivity axiom are the left sides of the first and the second of them, respectively. RL1 comes from Padmanabhan et al. [28] as an example of an equation which forces *the Huntington property* (distributivity under unique complementation).

AxRH1 is the strongest valid equation not derivable from AxRH2 (or other axioms in Table 1) that we were able to find. It can be seen as governing the interaction of natural join with projections. The axiom is surprisingly powerful; in particular, it will allow some simulation of quantifier behaviour and thus play a major rôle in our undecidability proof of quasiequational theory in Section 4. Equation Eq4 is one of its consequences which will also be useful in that proof.

$$\begin{tikzcd}
 & \mathrm{Qu1} & \mathrm{RL1} \arrow[l] & & \\
\mathrm{Qu2} \arrow[ur] \arrow[r, squiggly, no head] \arrow[ddr] & \cup \arrow[ur] \arrow[dr] \arrow[r, squiggly, no head] & \mathrm{Eq1} \arrow[r] & \mathrm{Eq2} \arrow[r] & \mathrm{Eq3} \\
 & & \mathrm{RL2} & & \\
 & \mathrm{AxRH2} \arrow[r, squiggly, no head] & \cup \arrow[uull] \arrow[d] & \mathrm{AxRH1} \arrow[l, squiggly, no head] \arrow[uul] \arrow[d] & \\
 & & \mathrm{Qu3} & \mathrm{Eq4} &
\end{tikzcd}$$

Figure 3: Dependencies between axioms in Table 1

Equation RL2 is rather surprising and intimidating-looking. Ironically, it is also the first valid axiom which does not hold in arbitrary lattices that we were able to find. Note that is obtained by substituting left and right sides of the obvious instances of one of the distributive laws in the following formula:

$$t_{\bowtie}(L(x,y,z) \oplus L(u,w,v)) = t_{\bowtie}(L(x,y,z) \oplus R(u,w,v)) \oplus t_{\bowtie}(R(x,y,z) \oplus L(u,w,v)),$$

RL2 is thus obtained by iterating and alternating two distributive laws.

Let us note the following dependencies between equations and quasiequations in Table 1:

**Theorem 3.4.** ***Assuming all lattice axioms***, *the following statements hold:*

1. *The axioms of $\underline{R}^{\mathsf{H}}$ in Table 1 are mutually independent.*
2. *The axioms of $\underline{R}$ are mutually independent.*
3. *RL1 implies Qu1 [28].*
4. *Eq1 implies Eq2.*
5. *Eq2 implies Eq3.*
6. *Qu2 together with Eq1 imply both RL1 and RL2.*
7. *Eq1 is implied by AxRH1. The converse implication does not hold even in presence of RL1.*
8. *AxRH1 and AxRH2 jointly imply Qu2, although each of the two equations separately is too weak to entail Qu2. In the converse direction, Qu2 implies AxRH2 but not AxRH1.*
9. *AxRH1 and AxRH2 jointly imply Qu3, although each of the two equations separately is too weak to entail Qu3 (in the case of AxRH2 even in presence of Eq1).*
10. *AxRH1 implies Eq4.*

*Proof.* Clause 1: For mutual independence of the two axioms of $\underline{R}^{\mathsf{H}}$, counterexamples can be obtained by appropriate choices of the interpretation of $\mathsf{H}$ in the pentagon lattice.
Clause 2: The example showing that the validity of RL2 does not imply the validity of RL1 is the non-distributive diamond lattice $M_3$, while the reverse implication can be disproved with an eight-element model:

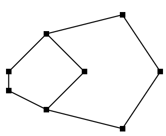

Table 1: (Quasi)equations Valid in Tropashko Lattices

Class $\underline{R}^{\mathsf{H}}$ in the signature $\mathcal{L}_{\mathsf{H}}$:

all lattice axioms

| | |
|---|---|
| AxRH1 | $\mathsf{H} \bowtie x \bowtie (y \oplus z) \oplus y \bowtie z = (\mathsf{H} \bowtie x \bowtie y \oplus z) \bowtie (\mathsf{H} \bowtie x \bowtie z \oplus y)$ |
| AxRH2 | $x \bowtie (y \oplus z) = x \bowtie (z \oplus \mathsf{H} \bowtie y) \oplus x \bowtie (y \oplus \mathsf{H} \bowtie z)$ |

Class $\underline{R}$ in the signature $\mathcal{L}$ (without $\mathsf{H}$):

all lattice axioms

| | |
|---|---|
| RL1 | $x \bowtie y \oplus x \bowtie z = x \bowtie (y \bowtie (x \oplus z) \oplus z \bowtie (x \oplus y))$ |
| RL2 | $t \bowtie ((x \oplus y) \bowtie (x \oplus z) \oplus (u \oplus w) \bowtie (u \oplus v)) =$ $= t \bowtie ((x \oplus y) \bowtie (x \oplus z) \oplus u \oplus w \bowtie v) \oplus t \bowtie ((u \oplus w) \bowtie (u \oplus v) \oplus x \oplus y \bowtie z)$ |

(in $\mathcal{L}_{\mathsf{H}}$, RL1 and RL2 are derivable from AxRH1 and AxRH2, see Theorem 3.4)

Additional (quasi)equations derivable in $\underline{R}^{\mathsf{H}}$ and $\underline{R}$:

| | | | |
|---|---|---|---|
| Qu1 | $x \oplus y = x \oplus z$ | $\Rightarrow$ | $x \bowtie (y \oplus z) = x \bowtie y \oplus x \bowtie z$ |
| Qu2 | $\mathsf{H} \bowtie (x \oplus y) = \mathsf{H} \bowtie (x \oplus z)$ | $\Rightarrow$ | $x \bowtie (y \oplus z) = x \bowtie y \oplus x \bowtie z$ |
| Qu3 | $\mathsf{H} \bowtie (x \oplus y) = \mathsf{H} \bowtie (x \oplus z) = \mathsf{H} \bowtie (y \oplus z)$ | $\Rightarrow$ | $x \oplus y \bowtie z = (x \oplus y) \bowtie (x \oplus z)$ |
| Eq1 | $\mathsf{H} \bowtie x \bowtie (y \oplus z)$ | $=$ | $\mathsf{H} \bowtie x \bowtie y \oplus \mathsf{H} \bowtie x \bowtie z$ |
| Eq2 | $\mathsf{H} \bowtie t \bowtie (x \oplus y) \bowtie (x \oplus z)$ | $=$ | $\mathsf{H} \bowtie t \bowtie x \oplus \mathsf{H} \bowtie t \bowtie y \bowtie z$ |
| Eq3 | $\mathsf{H} \bowtie (y \oplus z)$ | $=$ | $\mathsf{H} \bowtie y \oplus \mathsf{H} \bowtie z$ |
| Eq4 | $\mathsf{H} \bowtie x \oplus x \bowtie y$ | $=$ | $x \bowtie (\mathsf{H} \bowtie x \oplus y)$ |

Clause 4: We reason as follows:

$$\begin{aligned}
\mathsf{H}\bowtie t\bowtie(x\oplus y)\bowtie(x\oplus z) &= (\mathsf{H}\bowtie t\bowtie x\oplus \mathsf{H}\bowtie t\bowtie y)\bowtie(\mathsf{H}\bowtie t\bowtie x\oplus \mathsf{H}\bowtie t\bowtie z) && \text{Eq1, lattice laws}\\
&= \mathsf{H}\bowtie(t\bowtie x\oplus t\bowtie y)\bowtie(t\bowtie x\oplus t\bowtie z) && \text{Eq1, lattice laws}\\
&= \mathsf{H}\bowtie(t\bowtie x\oplus t\bowtie y)\bowtie t\bowtie x\oplus \mathsf{H}\bowtie(t\bowtie x\oplus t\bowtie y)\bowtie t\bowtie z && \text{Eq1}\\
&= \mathsf{H}\bowtie t\bowtie x\oplus \mathsf{H}\bowtie(t\bowtie x\oplus t\bowtie y)\bowtie t\bowtie z && \text{absorption}\\
&= \mathsf{H}\bowtie t\bowtie x\oplus \mathsf{H}\bowtie t\bowtie x\bowtie z\oplus \mathsf{H}\bowtie t\bowtie y\bowtie z && \text{Eq1, lattice laws}\\
&= \mathsf{H}\bowtie t\bowtie x\oplus \mathsf{H}\bowtie t\bowtie y\bowtie z && \text{lattice laws}
\end{aligned}$$

Clause 5: Substitute $\mathsf{H}$ for $t$, $z$ for $y$ and use lattice laws.
Clause 6: Direct computation. In more detail: for RL1, substitute $y\bowtie(x\oplus z)$ for $y$ and $z\bowtie(x\oplus y)$ for $z$ in the antecendent of Qu2. We get then the consequent of Qu2, as $\mathsf{H}\bowtie(x\oplus y\bowtie(x\oplus z)) = \mathsf{H}\bowtie x\oplus \mathsf{H}\bowtie y\bowtie(x\oplus z) = \mathsf{H}\bowtie x\oplus \mathsf{H}\bowtie y\bowtie x\oplus \mathsf{H}\bowtie y\bowtie z = \mathsf{H}\bowtie x\oplus \mathsf{H}\bowtie y\bowtie z = \mathsf{H}\bowtie x\oplus \mathsf{H}\bowtie x\bowtie z\oplus \mathsf{H}\bowtie y\bowtie z = \mathsf{H}\bowtie(x\oplus z\bowtie(x\oplus y))$ (we are obviously using Eq1 here). Thus, the right side of RL1 is equal to $x\bowtie y\bowtie(x\oplus z)\oplus x\bowtie z\bowtie(x\oplus y)$. But this, by the absorption law, is equal to $x\bowtie y\oplus x\bowtie z$, i.e., the left side of RL1.

For the seemingly monstrous RL2, the trick is similar. Consider

$$\begin{aligned}
\mathsf{H}\bowtie((x\oplus y)\bowtie(x\oplus z)\oplus u\oplus w\bowtie v) &= \mathsf{H}\bowtie(x\oplus y)\bowtie(x\oplus z)\oplus \mathsf{H}\bowtie u\oplus \mathsf{H}\bowtie w\bowtie v\\
&= \mathsf{H}\bowtie x\oplus \mathsf{H}\bowtie y\bowtie z\oplus \mathsf{H}\bowtie(u\oplus w)\bowtie(u\oplus v)\\
&= \mathsf{H}\bowtie(x\oplus y\bowtie z)\oplus \mathsf{H}\bowtie(u\oplus w)\bowtie(u\oplus v)\\
&= \mathsf{H}\bowtie((u\oplus w)\bowtie(u\oplus v)\oplus(x\oplus y\bowtie z))
\end{aligned}$$

and thus

$$\mathsf{H}\bowtie(t\oplus(x\oplus y)\bowtie(x\oplus z)\oplus u\oplus w\bowtie v) = \mathsf{H}\bowtie(t\oplus(u\oplus w)\bowtie(u\oplus v)\oplus(x\oplus y\bowtie z))$$

This allows us to use Qu2 to rewrite the right side of RL2:

$$\begin{aligned}
&t\bowtie((x\oplus y)\bowtie(x\oplus z)\oplus u\oplus w\bowtie v)\oplus t\bowtie((u\oplus w)\bowtie(u\oplus v)\oplus x\oplus y\bowtie z)\\
&\qquad= t\bowtie((x\oplus y)\bowtie(x\oplus z)\oplus u\oplus w\bowtie v\oplus(u\oplus w)\bowtie(u\oplus v)\oplus x\oplus y\bowtie z)\\
&\qquad= t\bowtie((x\oplus y)\bowtie(x\oplus z)\oplus(u\oplus w)\bowtie(u\oplus v))
\end{aligned}$$

(the second equality obtaining by lattice laws).
Clause 7: The first part has been proved with the help of Prover9 (66 lines of proof—see Appendix). The counterexample for the converse has been found by Mace4: it is obtained by choosing $\mathsf{H}$ to be the top element of the pentagon lattice.
Clause 8: Prover9 was able to prove the first statement both in presence and in absence of RL1, although there was a significant difference in the length of both proofs (38 lines vs. 195 lines—see Appendix). The implication from Qu2 to AxRH2 is straightforward. All the necessary counterexamples have been found by Mace4 by appropriate choices of the interpretation of $\mathsf{H}$ in the pentagon lattice.
Clause 9: The positive statement was proved by Prover9 (mere 196 lines—see Appendix). Again, counterexamples for all the negative statements can be found using 5-element models.
Clause 10: Substitue $x$ for $z$ and use the absorption law. □

**Theorem 3.5.** *The following equivalent statements hold:*

- *AxRH1 and AxRH2 are valid in $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ (and consequently in $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$).*
- *AxRH1, AxRH2 and Eq1 are valid in $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ (and consequently in $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$).*
- *Axioms of $\underline{R}^{\mathsf{H}}$ are valid in $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ (and consequently in $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$). Similarly, axioms of $\underline{R}$ are valid in $\mathcal{R}_{\mathrm{unr}}$ (and consequently $\mathcal{R}_{\mathrm{fin}}$).*

- *All formulas in Table 1 are valid in* $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ *(and consequently in* $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$*). Those not involving* $\mathsf{H}$ *are valid in* $\mathcal{R}_{\mathrm{unr}}$ *(and consequently in* $\mathcal{R}_{\mathrm{fin}}$*).*

*Proof.* Theorem 3.4 implies that all clauses are equivalent, so we can choose whichever we want to prove. Perhaps the most convenient is the second one. Validity of Eq1 is immediate, as the sublattice of relations with empty body (we can call it the *header sublattice*) is obviously distributive. In presence of Eq1, we obtain automatically

$$\mathsf{H}\bowtie x\bowtie(y\oplus z)\oplus y\bowtie z \leq (\mathsf{H}\bowtie x\bowtie y\oplus z)\bowtie(\mathsf{H}\bowtie x\bowtie z\oplus y),$$

so to establish AxRH1, it is enough to establish the other inequality. Denote $H_{x\bowtie y}\cap H_z$ as $H_1$ and $H_{x\bowtie z}\cap H_y$ as $H_2$; note that $H_1\cap H_2 = H_y\cap H_z$. A tuple $t$ belongs to the body of $(\mathsf{H}\bowtie x\bowtie y\oplus z)\bowtie(\mathsf{H}\bowtie x\bowtie z\oplus y)$ iff there exists $t_1\in B_z[H_1]$ and $t_2\in B_y[H_2]$ s.t. $t_1[H_1\cap H_2] = t_2[H_1\cap H_2]$ and $t$ is the concatenation of $t_1$ and $t_2$. This is in turn equivalent to the existence of $t'_1\in B_z$ and $t'_2\in B_y$ s.t. $t'_1[H_1\cap H_2] = t'_2[H_1\cap H_2]$ ($t_1$ being $t'_1[H_1]$ and $t_2 = t'_2[H_2]$); by our earlier observation, $H_1\cap H_2$ is precisely the set of attributes on which headers of $t'_1$ and $t'_2$ overlap, so we can see $t$ as the restriction of concatenation of $t'_1$ and $t'_2$ to $H_1\cup H_2$. But this means that $t$ belongs to the body of $\mathsf{H}\bowtie x\bowtie(y\oplus z)\oplus y\bowtie z$.

Finally, let us consider AxRH2. Lattice laws yield that

$$x\bowtie(y\oplus z)\geq x\bowtie(z\oplus\mathsf{H}\bowtie y)\oplus x\bowtie(y\oplus\mathsf{H}\bowtie z),$$

so we only need to establish the opposite inequality. Pick any $t$ in the body of $x\bowtie(y\oplus z)$. Clearly, there exist $t_x\in B_x$ and $t_2\in B_{y\oplus z}$ overlapping on $H_x\cap H_y\cap H_z$ and $t$ is their concatenation. Now, $t_2$ is either a restriction of some $t_y\in B_y$ or of some $t_z\in B_z$. Assume the first case; we get then that $t_2$ belongs to the body of $y\oplus\mathsf{H}\bowtie z$ and consequently $t\in x\bowtie(y\oplus\mathsf{H}\bowtie z)$. Similarly, in the other case we get that $t\in x\bowtie(z\oplus\mathsf{H}\bowtie y)$. $\square$

**Open Problem 3.6.** *Are the equational theories of* $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ *(resp.* $\mathcal{R}_{\mathrm{unr}}$*) and* $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$ *(resp.* $\mathcal{R}_{\mathrm{fin}}$*) equal? How about quasiequational ones?*

**Open Problem 3.7.** *Is the equational theory of* $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ *(resp.* $\mathcal{R}_{\mathrm{unr}}$*) equal to* $\underline{R}^{\mathsf{H}}$ *(*$\underline{R}$*, respectively)? If not, is it finitely axiomatizable at all?*

If the answer to the last question is in the negative, one can perhaps attempt a rainbow-style argument from algebraic logic [16].

**Remark 3.8.** *When completing the final version of this paper, we learned of the recent work of Santocanale [33] building on it. Its most important contribution consists in finding a number of additional valid equalities in pure lattices signature (i.e.,* $\mathcal{L}$*) not derivable from RL1 and RL2. This shows that these two equalities are not sufficient or suitable as axioms for the abstract class of relational lattices in signature* $\mathcal{L}$ *and hence we dropped the prefix "Ax" in this case. However, all equalities found by Santocanale are derivable in signature* $\mathcal{L}_{\mathsf{H}}$ *from AxRH1 and AxRH2 and these equations remain the best candidates for axioms that we have. The question of complete axiomatization for either* $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ *or* $\mathcal{R}_{\mathrm{unr}}$ *remains open. We have also obtained a decidability proof which will be published elsewhere.*

## 4. Relational Lattices as a Quasiequational Class

In the introduction, we discussed why an axiomatization of valid *quasi*equations is desirable from a DB point of view. There is also an algebraic reason: the class of representable Tropashko lattices (i.e., the $\mathbb{SP}$-closure of concrete ones) is a *quasi*variety. This is a corollary of a more powerful result; recall that being *pseudoelementary* means being a reduct of an elementary class in a richer (possibly multi-sorted) language and that this notion plays a central rôle in algebraic study of axiomatizability and representability [16]:

**Theorem 4.1.** $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ *and* $\mathcal{R}_{\mathrm{unr}}$ *are pseudoelementary classes.*

*Proof.* (sketch)

We show that relational lattices form a pseudo-axiomatizable class: there is a finite set of first-order axioms $Ax$ such that for any first-order structure $\mathfrak{M}$ of the appropriate signature,

$$\mathfrak{M} \models Ax \text{ iff the } (R, \bowtie, \oplus)\text{-reduct of } \mathfrak{M} \text{ is a relational lattice.}$$

$\mathfrak{M}$ will have the following sorts: $A, D, P, S, H, B, R$ with the intended interpretations $A \subseteq \mathcal{A}, D \subseteq \mathcal{D}, P \subseteq \mathcal{A} \times \mathcal{D}$,

$$S \subseteq \{f \mid f \text{ is a function with } \mathrm{dom}(f) \subseteq \mathcal{A}, \mathrm{ran}(f) \subseteq \mathcal{D}\},$$

$H \subseteq \wp(\mathcal{A})$ is closed under union and intersection, $B \subseteq \wp(S)$ such that every $b \in B$ is the body of a relation, and $R \subseteq H \times B$ is closed under the operations $\bowtie$ and $\oplus$ (see below), respectively.

$\mathfrak{M}$ will have the following functions: $\mathsf{left}\colon P \to A$, $\mathsf{right}\colon P \to D$, $\cup\colon A \times A \to A$, $\cap\colon A \times A \to A$, $\mathsf{head}\colon S \to H$, $\restriction\colon S \times H \to S$, $\mathsf{Head}\colon R \to H$, $\mathsf{Body}\colon R \to B$, $\bowtie\colon R \times R \to R$, $\oplus\colon R \times R \to R$ and relation: $\in\colon (P \times S) \cup (A \times H) \cup (S \times B)$.

Then $Ax$ is defined as the collection of the following fourteen axioms.

Elements of $P$ are ordered pairs from $A \times D$:

$$(\forall p_1, p_2 \in P)((\mathsf{left}(p_1) = \mathsf{left}(p_2) \wedge \mathsf{right}(p_1) = \mathsf{right}(p_2)) \to p_1 = p_2) \tag{5}$$

Elements of $S$ are partial functions from $A$ to $D$:

$$(\forall s_1, s_2 \in S)((\forall p \in P)(p \in s_1 \leftrightarrow p \in s_2) \to s_1 = s_2) \tag{6}$$

$$(\forall s \in S)(\forall p_1, p_2 \in s)(\mathsf{left}(p_1) = \mathsf{left}(p_2) \to \mathsf{right}(p_1) = \mathsf{right}(p_2)) \tag{7}$$

Elements of $H$ are subsets of $A$ augmented with union and intersection:

$$(\forall h_1, h_2 \in H)((\forall a \in A)(a \in h_1 \leftrightarrow a \in h_2) \to h_1 = h_2) \tag{8}$$

$$(\forall h_1, h_2 \in H)(\forall a \in A)(a \in h_1 \cup h_2 \leftrightarrow (a \in h_1 \vee a \in h_2)) \tag{9}$$

$$(\forall h_1, h_2 \in H)(\forall a \in A)(a \in h_1 \cap h_2 \leftrightarrow (a \in h_1 \wedge a \in h_2)) \tag{10}$$

The function $\mathsf{head}$ gives the domains of elements of $S$:

$$(\forall s \in S)(\forall a \in A)(a \in \mathsf{head}(s) \leftrightarrow (\exists p \in s)\mathsf{left}(p) = a) \tag{11}$$

The function $\restriction$ gives the restrictions of elements of $S$ to headers in $H$:

$$\begin{aligned}(\forall s_1, s_2 \in S)(\forall h \in H)(s_1 \restriction h = s_2 \leftrightarrow \\ (\mathsf{head}(s_2) = h \wedge (\forall p \in P)(\mathsf{left}(p) \in h \to (p \in s_1 \leftrightarrow p \in s_2)))\end{aligned} \tag{12}$$

Bodies of elements of $R$ are subsets of $S$ with the same domain:

$$(\forall r_1, r_2 \in R)(\forall s \in S)((s \in \mathsf{Body}(r_1) \leftrightarrow s \in \mathsf{Body}(r_2)) \to \mathsf{Body}(r_1) = \mathsf{Body}(r_2)) \tag{13}$$

$$(\forall r \in R)(\forall s_1, s_2 \in \mathsf{Body}(r))\mathsf{head}(s_1) = \mathsf{head}(s_2) \tag{14}$$

We extend the restriction operation to bodies of relations by

$$\mathsf{Body}(r) \restriction h = \{s \restriction h \mid s \in \mathsf{Body}(r)\}$$

for every $r \in R$ and $h \in H$.

Relations are pairs $(h, b)$ of matching headers and bodies:

$$(\forall r_1, r_2 \in R)((\mathsf{Head}(r_1) = \mathsf{Head}(r_2) \wedge \mathsf{Body}(r_1) = \mathsf{Body}(r_2)) \to r_1 = r_2) \tag{15}$$

$$(\forall r \in R)\mathsf{Body}(r) \restriction \mathsf{Head}(r) = \mathsf{Body}(r) \tag{16}$$

Definition of $\bowtie$:

$$\begin{aligned}(\forall r_1, r_2 \in R)(\forall s \in S)(s \in \mathsf{Body}(r_1 \bowtie r_2) \leftrightarrow (\mathsf{head}(s) = \mathsf{Head}(r_1) \cup \mathsf{Head}(r_2) \wedge \\ s \restriction \mathsf{Head}(r_1) \in \mathsf{Body}(r_1) \wedge s \restriction \mathsf{Head}(r_2) \in \mathsf{Body}(r_2)))\end{aligned} \tag{17}$$

Definition of $\oplus$:

$$(\forall r_1, r_2 \in R)(\forall s \in S)(s \in \mathsf{Body}(r_1 \oplus r_2) \leftrightarrow (\mathsf{head}(s) = \mathsf{Head}(r_1) \cap \mathsf{Head}(r_2) \wedge \\ (s \in \mathsf{Body}(r_1) \restriction \mathsf{Head}(r_2) \vee s \in \mathsf{Body}(r_2) \restriction \mathsf{Head}(r_1)))) \tag{18}$$

Assume $\mathfrak{M} \models Ax$. Define the map

$$r \mapsto (\{a \in A \mid a \in \mathsf{Head}(r)\}, \{s \in S \mid s \in \mathsf{Body}(r)\})$$

(for every $r \in R$) and define $\bowtie$ and $\oplus$ as in the definition of relational lattices. Then it is straightforward to check that the $(R, \bowtie, \oplus)$-reduct of $\mathfrak{M}$ can be isomorphically embedded via $\mapsto$ into $\mathfrak{R}(D, A)$.

For a pseudo-axiomatization of $\mathfrak{R}^{\mathsf{H}}$ we have the additional requirement that $\mathfrak{M}$ contains the constant $\mathsf{H}$ of sort $R$ satisfying the additional axiom

$$(\forall a \in A)(\forall s \in S)(a \notin \mathsf{Head}(\mathsf{H}) \wedge s \notin \mathsf{Body}(\mathsf{H})) \tag{19}$$

ensuring that $\mathsf{H}$ can be mapped to the header constant $(\emptyset, \emptyset)$.

□

**Corollary 4.2.** *$\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ and $\mathcal{R}_{\mathrm{unr}}$ are closed under ultraproducts.*

**Corollary 4.3.** *The $\mathbb{SP}$-closures of $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ and $\mathcal{R}_{\mathrm{unr}}$ are quasiequational classes.*

**Corollary 4.4.** *The quasiequational, universal and elementary theories of $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ and $\mathcal{R}_{\mathrm{unr}}$ are recursively enumerable.*

*Proof.* The proof of Theorem 4.1 uses finitely many axioms. □

Note that postulating that headers are *finite* subsets of $\mathcal{A}$ would break the proof of Theorem 4.1: such conditions are not first-order. However, concrete database instances always belong to $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$ and we will show now that the decidability status of the quasiequational theories of $\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}$ and $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$ is the same. Moreover, corresponding abstract classes also have undecidable quasiequational theories, much like for relation algebras and cylindric algebras—in fact, we build on a proof of Maddux [25] for $CA_3$—and we *do not even need all the axioms* of $\underline{R}^{\mathsf{H}}$ to show this! Let $\underline{RH1}$ be the variety of $\mathcal{L}_{\mathsf{H}}$-algebras axiomatized by the lattice axioms and AxRH1. Let us list some basic observations:

**Proposition 4.5.**

1. $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}} \subset \mathcal{R}^{\mathsf{H}}_{\mathrm{unr}} \subset \mathbb{SP}(\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}}) \subseteq \underline{R}^{\mathsf{H}} \subset \underline{RH1}$.
2. *Eq4 holds in $\underline{RH1}$.*
3. *AxRH1 holds whenever $\mathsf{H}$ is interpreted as the bottom of a bounded lattice.*
4. *AxRH1 holds for an arbitrary choice of $\mathsf{H}$ in a distributive lattice.*

*Proof.* Clause 2 holds by clause 10 of Theorem 3.4. The remaining ones are straightforward to verify. □

Note, e.g., that interpreting $\mathsf{H}$ as $\perp$ in AxRH2 would only work if the lattice is distributive, so Clause 3 does not hold in general for AxRH2. In order to state our undecidability result, we need first

**Definition 4.6.** *Let $\overline{e} = (u_0, u_1, u_2, e_0, e_1)$ be an arbitrary 5-tuple of variables. We abbreviate $u_0 \bowtie u_1 \bowtie u_2$ as $u$. For arbitrary terms $s, t$ define the following syntactic abbreviations, which we use to simulate quantification and to imitate the monoid operation just like it is done in cylindric algebras [25]:*

$$\begin{aligned}
\mathbf{c}^{\overline{e}}_0 \langle s \rangle &:= u \bowtie (\mathsf{H} \bowtie u_1 \bowtie u_2 \oplus u \bowtie s), \\
\mathbf{c}^{\overline{e}}_1 \langle s \rangle &:= u \bowtie (\mathsf{H} \bowtie u_0 \bowtie u_2 \oplus u \bowtie s), \\
\mathbf{c}^{\overline{e}}_2 \langle s \rangle &:= u \bowtie (\mathsf{H} \bowtie u_0 \bowtie u_1 \oplus u \bowtie s), \\
s \circ^{\overline{e}} t &:= \mathbf{c}^{\overline{e}}_2 \left\langle \mathbf{c}^{\overline{e}}_1 \left\langle e_0 \bowtie \mathbf{c}^{\overline{e}}_2 \langle s \rangle \right\rangle \bowtie \mathbf{c}^{\overline{e}}_0 \left\langle e_1 \bowtie \mathbf{c}^{\overline{e}}_2 \langle t \rangle \right\rangle \right\rangle.
\end{aligned}$$

*Let $T_n(x_1, \ldots, x_n)$ be the collection of all semigroup terms in $n$ variables. Pick $\overline{e} := (x_{n+1}, \ldots, x_{n+5})$ and define the translation $\tau^{\overline{e}} : T_n(x_1, \ldots, x_n) \to \mathcal{L}_{\mathsf{H}}$ as follows: $\tau^{\overline{e}}(x_i) := x_i$ for $i \leq n$ and $\tau^{\overline{e}}(s \circ t) := \tau^{\overline{e}}(s) \circ^{\overline{e}} \tau^{\overline{e}}(t)$ for any $s, t \in T_n(x_1, \ldots, x_n)$.*

Whenever $\overline{e}$ is clear from the context, we will drop it to ensure readability. Now we can formulate

**Theorem 4.7.** *For any $p_0, \ldots, p_m, r_0, \ldots, r_m, s, t \in T_n(x_1, \ldots, x_n)$, the following conditions are equivalent:*

*(I) The quasiequation*

$$\text{(Qu4)} \qquad \forall x_1, \ldots, x_n . \, (p_0 = r_0 \,\&\, \ldots \,\&\, p_m = r_m \Rightarrow s = t)$$

*holds in all semigroups (resp. finite semigroups).*

*(II) For $\overline{e} = (x_{n+1}, \ldots, x_{n+5})$ as in Definition 4.6, the quasiequation*

$$\text{(Qu5)} \qquad \begin{aligned} \forall x_0, x_1, \ldots, x_{n+5} . \, (\tau^{\overline{e}}(p_0) = \tau^{\overline{e}}(r_0) \,\&\, \ldots \tau^{\overline{e}}(p_m) = \tau^{\overline{e}}(r_m) \,\&\, \\ \&\, x_{n+4} = \mathbf{c}_0^{\overline{e}} \langle x_{n+4} \rangle \,\&\, x_{n+5} = \mathbf{c}_1^{\overline{e}} \langle x_{n+5} \rangle) \Rightarrow \\ \Rightarrow \tau^{\overline{e}}(s) \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}} \langle x_0 \rangle = \tau^{\overline{e}}(t) \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}} \langle x_0 \rangle)) \end{aligned}$$

*holds in every member of $\mathcal{R}^{\mathsf{H}}_{\text{unr}}$ (resp. every member of $\mathcal{R}^{\mathsf{H}}_{\text{fin}}$).*

*(III) Qu5 above holds in every member of $\underline{RH1}$ (resp. finite member of $\underline{RH1}$).*

*Proof.* (I) $\Rightarrow$ (III). By contraposition:

Take any $\mathfrak{A} \in \underline{RH1}$ and arbitrarily chosen elements $u_0, u_1, u_2 \in \mathfrak{A}$. In order to use Maddux's technique, we have to prove that for any $a, b \in \mathfrak{A}$ and $k, l < 3$

(b) $\mathbf{c}_k \langle \mathbf{c}_k \langle a \rangle \rangle = \mathbf{c}_k \langle a \rangle$,
(c) $\mathbf{c}_k \langle a \ltimes \mathbf{c}_k \langle b \rangle \rangle = \mathbf{c}_k \langle a \rangle \ltimes \mathbf{c}_k \langle b \rangle$,
(d) $\mathbf{c}_k \langle \mathbf{c}_l \langle a \rangle \rangle = \mathbf{c}_l \langle \mathbf{c}_k \langle a \rangle \rangle$

(we deliberately keep the same labels as in the quoted paper), where $\mathbf{c}_k \langle a \rangle$ is defined in the same way as in Definition 4.6 above. We will denote by $u[\hat{k}]$ the product of $u_i$'s such that $i \in \{0, 1, 2\} - \{k\}$. For example, $u[\hat{0}] = u_1 \ltimes u_2$.

For (b):

$$\begin{aligned} \mathbf{c}_k \langle \mathbf{c}_k \langle a \rangle \rangle &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a)) \\ &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \ltimes (u \oplus \mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a) \oplus u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a)) && \text{by lattice laws} \\ &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \ltimes u \oplus \mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a) \ltimes (\mathsf{H} \ltimes u[\hat{k}] \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a) \oplus u) && \text{by AxRH1} \\ &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a) \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u) && \text{by lattice laws} \\ &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a) && \text{by lattice laws} \\ &= \mathbf{c}_k \langle a \rangle . \end{aligned}$$

(c) is proved using a similar trick:

$$\begin{aligned} \mathbf{c}_k \langle a \ltimes \mathbf{c}_k \langle b \rangle \rangle &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes b)) \\ &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \ltimes (u \ltimes a \oplus \mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes b) \oplus u \ltimes a \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes b)) && \text{by lattice laws} \\ &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \ltimes u \ltimes a \oplus \mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes b) \ltimes (\mathsf{H} \ltimes u[\hat{k}] \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes b) \oplus u \ltimes a) && \text{by AxRH1} \\ &= u \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes b) \ltimes (\mathsf{H} \ltimes u[\hat{k}] \oplus u \ltimes a) && \text{by lattice laws} \\ &= \mathbf{c}_k \langle a \rangle \ltimes \mathbf{c}_k \langle b \rangle . \end{aligned}$$

(d) is obviously true for $k = l$, hence we can restrict attention to $k \neq l$. Let $j$ be the remaining element of $\{0, 1, 2\}$. Thus,

$$
\begin{aligned}
\mathbf{c}_k\left\langle \mathbf{c}_l\left\langle a\right\rangle\right\rangle &= u⋉(\mathsf{H}⋉u_l⋉u_j \oplus u⋉(\mathsf{H}⋉u_k⋉u_j \oplus u⋉a)) \\
&= u⋉(\mathsf{H}⋉u_l⋉u_j \oplus u_l⋉u_k⋉u_j⋉(\mathsf{H}⋉u_k⋉u_j \oplus u_k⋉u_j⋉u_l⋉a)) && \text{by definition of } u \\
&= u⋉(\mathsf{H}⋉u_l⋉u_j \oplus u_l⋉(\mathsf{H}⋉u_k⋉u_j \oplus u⋉a)) && \text{by lattice laws and Eq4} \\
&= u⋉(\mathsf{H}⋉u_l⋉u_j⋉(u_l \oplus \mathsf{H}⋉u_k⋉u_j \oplus u⋉a) \oplus u_l⋉(\mathsf{H}⋉u_k⋉u_j \oplus u⋉a)) && \text{by lattice laws} \\
&= u⋉(\mathsf{H}⋉u_l⋉u_j \oplus \mathsf{H}⋉u_k⋉u_j \oplus u⋉a)⋉(\mathsf{H}⋉u_l⋉u_j⋉(\mathsf{H}⋉u_k⋉u_j \oplus u⋉a) \oplus u_l) && \text{by AxRH1} \\
&= u⋉(\mathsf{H}⋉u_l⋉u_j \oplus \mathsf{H}⋉u_k⋉u_j \oplus u⋉a)⋉u_l && \text{by lattice laws} \\
&= u⋉(\mathsf{H}⋉u_l⋉u_j \oplus \mathsf{H}⋉u_k⋉u_j \oplus u⋉a) && \text{by lattice laws}
\end{aligned}
$$

and in the last term, $u_l$ and $u_k$ may be permuted by commutativity. We then obtain the right side of the equation via an analogous sequence of transformations in the reverse direction, with the rôles of $u_k$ and $u_l$ replaced.

The rest of the proof mimics the one by Maddux [25]. In some detail[3]: assume there is

$$\overline{e} = (u_0, u_1, u_2, e_0, e_1) \in \mathfrak{A}^5$$

such that

(a) $\mathbf{c}_0^{\overline{e}}\left\langle e_0\right\rangle = e_0$, $\mathbf{c}_1^{\overline{e}}\left\langle e_1\right\rangle = e_1$

holds. Using (a)–(d) we prove that for every $a, b \in \mathfrak{A}$ the following hold:

(i) $\mathbf{c}_1^{\overline{e}}\left\langle a \circ^{\overline{e}} b\right\rangle = a \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle b\right\rangle$,
(ii) $a \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle b\right\rangle = \mathbf{c}_1^{\overline{e}}\left\langle \mathbf{c}_2^{\overline{e}}\left\langle a\right\rangle ⋉ \mathbf{c}_0^{\overline{e}}\left\langle \mathbf{c}_2^{\overline{e}}\left\langle e_0⋉e_1⋉\mathbf{c}_2^{\overline{e}}\left\langle \mathbf{c}_1^{\overline{e}}\left\langle b\right\rangle\right\rangle\right\rangle\right\rangle\right\rangle$,
(iii) $(a \circ^{\overline{e}} b) \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle c\right\rangle = a \circ^{\overline{e}} (b \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle c\right\rangle)$,
(iv) $((a \circ^{\overline{e}} b) \circ^{\overline{e}} c) \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle d\right\rangle = (a \circ^{\overline{e}} (b \circ^{\overline{e}} c)) \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle d\right\rangle$.

Now pick $\mathfrak{A}$ witnessing the failure of Qu5 together with $\overline{e} = (u_0, u_1, u_2, e_0, e_1)$ such that elements of $\overline{e}$ interpret variables $(x_{n+1}, \ldots, x_{n+5})$ in Qu5. This means (a) is satisfied, hence (i)–(iv) hold for every element of $\mathfrak{A}$. We define an equivalence relation $\equiv$ on $\mathfrak{A}$:

$a \equiv b$ iff for all $c \in \mathfrak{A}$, $a \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle c\right\rangle = b \circ^{\overline{e}} \mathbf{c}_1^{\overline{e}}\left\langle c\right\rangle$.

We take $\circ^{\overline{e}}$ to be the semigroup operation on $\mathfrak{A}/\equiv$. Following Maddux [25], we use (i)–(iv) to prove that this operation is well-defined (i.e., independent of the choice of representatives) and satisfies semigroup axioms. It follows from the assumptions that the semigroup thus defined fails Qu4.

(III) $\Rightarrow$ (II). Immediate.

(II) $\Rightarrow$ (I). In analogy to Maddux [25], given a semigroup $\mathfrak{B} = (B, \circ, \mathsf{u})$ failing Qu4 and a valuation $v$ witnessing this failure, consider $\mathfrak{R}(B, \{0,1,2\})$ with a valuation $w$ defined as follows:

$$
\begin{aligned}
w(x_0) &:= (\{0,1,2\}, \{\{(0, v(s)), (1, a), (2, b)\} \mid a, b \in \mathfrak{B}\}), \\
w(x_i) &:= (\{0,1,2\}, \{\{(0, a), (1, a \circ v(x_i)), (2, b)\} \mid a, b \in \mathfrak{B}\}), && i \leq n, \\
w(x_{n+i}) &:= (\{i\}, \{\{(i, b)\} \mid b \in \mathfrak{B}\}), && (0 < i \leq 3), \\
w(x_{n+4}) &:= (\{0,1,2\}, \{\{(0, a), (1, b), (2, b)\} \mid a, b \in \mathfrak{B}\}), \\
w(x_{n+5}) &:= (\{0,1,2\}, \{\{(0, b), (1, a), (2, b)\} \mid a, b \in \mathfrak{B}\}).
\end{aligned}
$$

[3] We hope the reader is not confused by our use of the same symbols for elements of algebra as earlier for terms and variables; an alternative relying on changing all the notations could prove worse.

It may be proved by induction that

$$w(\tau^{\overline{e}}(u)) = (\{0,1,2\}, \{\{(0,a),(1,a\circ v(u)),(2,b)\} \mid a,b\in\mathfrak{B}\})$$

(where $e = (x_{n+1},\dots,x_{n+5})$) for every $u \in T(x_1,\dots x_n)$. Moreover

$$\begin{aligned}
w(\tau^{\overline{e}}(t)\circ^{\overline{e}}\mathbf{c}_1^{\overline{e}}\langle x_0\rangle) =&(\{0,1,2\}, \{\{(0,a),(1,b),(2,c)\} \mid a,b,c\in\mathfrak{B}, a\circ v(t)=v(s)\}),\\
w(\tau^{\overline{e}}(s)\circ^{\overline{e}}\mathbf{c}_1^{\overline{e}}\langle x_0\rangle) =&(\{0,1,2\}, \{\{(0,a),(1,b),(2,c)\} \mid a,b,c\in\mathfrak{B}, a\circ v(s)=v(s)\}).
\end{aligned}$$

Any tuple whose value for attribute 0 is $\mathsf{u}$ belongs to the first relation, but not to the second. Thus $w$ is a valuation refuting Qu5. □

**Corollary 4.8.** *The quasiequational theory of any class of algebras between $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$ and $\underline{RH1}$ is undecidable.*

*Proof.* Follows from Theorem 4.7 and theorems of Gurevič [13, 14] and Post [29] (for finite and arbitrary semigroups, respectively). □

**Corollary 4.9.** *The quasiequational theory of $\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}}$ is not finitely axiomatizable.*

*Proof.* Follows from Theorem 4.7 and the Harrop criterion [15]. □

**Open Problem 4.10.** *Are the quasiequational theories of $\mathcal{R}_{\mathrm{unr}}$ and $\mathcal{R}_{\mathrm{fin}}$ (i.e., of lattice reducts) decidable?*

## 5. The Concept Structure of Tropashko Lattices

Given a finite lattice $\mathcal{L}$ with $\mathfrak{J}(\mathcal{L})$ and $\mathfrak{M}(\mathcal{L})$ being the sets of its, respectively, join- and meet-irreducible elements, let us follow Formal Concept Analysis [12] and investigate the structure of $\mathcal{L}$ via its *standard context* $\mathsf{con}(\mathcal{L}) := (\mathfrak{J}(\mathcal{L}), \mathfrak{M}(\mathcal{L}), \mathsf{I}_{\leq})$, where $\mathsf{I}_{\leq} := \leq \cap (\mathfrak{J}(\mathcal{L})\times\mathfrak{M}(\mathcal{L}))$. Set

$$\begin{aligned}
g \swarrow m :&\quad g \text{ is } \leq\text{-minimal in } \{h\in\mathfrak{J}(\mathcal{L}) \mid \text{not } h\,\mathsf{I}_{\leq}\, m\},\\
g \nearrow m :&\quad m \text{ is } \leq\text{-maximal in } \{n\in\mathfrak{M}(\mathcal{L}) \mid \text{not } g\,\mathsf{I}_{\leq}\, n\},\\
g \swarrow\!\!\!\nearrow m :&\quad g\swarrow m \,\&\, g\nearrow m.
\end{aligned}$$

Let also $\swarrow\!\!\swarrow$ be the smallest relation containing $\swarrow$ and satisfying the condition

$$g \swarrow\!\!\swarrow m,\ h\nearrow m \text{ and } h\swarrow n \text{ imply } g\swarrow\!\!\swarrow n;$$

in a more compact notation, $\swarrow\!\!\swarrow \circ \nearrow \circ \swarrow \subseteq \swarrow\!\!\swarrow$. We have the following

**Proposition 5.1.** *[12, Theorem 17] A finite lattice is*

- *subdirectly irreducible iff there is $m\in\mathfrak{M}(\mathcal{L})$ such that $\swarrow\!\!\swarrow \supseteq \mathfrak{J}(\mathcal{L})\times\{m\}$,*
- *simple iff $\swarrow\!\!\swarrow = \mathfrak{J}(\mathcal{L})\times\mathfrak{M}(\mathcal{L})$.*

Let us describe $\mathfrak{J}(\mathfrak{R}(\mathcal{D},\mathcal{A}))$ and $\mathfrak{M}(\mathfrak{R}(\mathcal{D},\mathcal{A}))$ for finite $\mathcal{D}$ and $\mathcal{A}$. Set

$$\begin{array}{llll}
\mathcal{AD}om_{\mathcal{D},\mathcal{A}} & := \{\mathsf{adom}(x) \mid x\in\mathcal{D}^{\mathcal{A}}\} & \text{where } \mathsf{adom}(x) & := (\mathcal{A},\{x\}),\\
\mathcal{AA}tt_{\mathcal{D},\mathcal{A}} & := \{\mathsf{aatt}(a) \mid a\in\mathcal{A}\} & \text{where } \mathsf{aatt}(a) & := (\mathcal{A}-\{a\},\emptyset),\\
\mathcal{C}oDom_{\mathcal{D},H} & := \{\mathsf{codom}^H(x) \mid x\in\mathcal{D}^H\} & \text{where } \mathsf{codom}^H(x) & := (H,\mathcal{D}^H-\{x\}),\\
\mathcal{C}oAtt_{\mathcal{D},\mathcal{A}} & := \{\mathsf{coatt}(a) \mid a\in\mathcal{A}\} & \text{where } \mathsf{coatt}(a) & := (\{a\},\mathcal{D}^{\{a\}}),
\end{array}$$

$$\begin{array}{lll}
\mathcal{J}_{\mathcal{D},\mathcal{A}} & := & \mathcal{AD}om_{\mathcal{D},\mathcal{A}} \cup \mathcal{AA}tt_{\mathcal{D},\mathcal{A}},\\
\mathcal{M}_{\mathcal{D},\mathcal{A}} & := & \mathcal{C}oAtt_{\mathcal{D},\mathcal{A}} \cup \bigcup_{H\subseteq\mathcal{A}} \mathcal{C}oDom_{\mathcal{D},H}.
\end{array}$$

It is worth noting that $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ naturally divides into what we may call *boolean H-fibres*—i.e., the powerset algebras of $\mathcal{D}^H$ for each $H \subseteq \mathcal{A}$. Furthermore, the projection mapping from $H$-fibre to $H'$-fibre where $H' \subseteq H$ is a join-homomorphism. Lastly, note that the bottom elements of $H$-fibres—i.e., elements of the form $(H, \emptyset)$—and top elements of the form $(H, \mathcal{D}^H)$ form two additional boolean slices, which we may call the *lower attribute slice* and *the upper attribute slice*, respectively. Both slices are obviously isomorphic copies of the powerset algebra of $\mathcal{A}$. The intention of our definition should be clear then:

- The join-irreducibles are only the atoms of the $\mathcal{A}$-fibre (i.e., the fibre with the longest tuples) plus the atoms of the lower attribute slice.
- The set of meet-irreducibles is much richer: it consists of the coatoms of *all H-fibres* (note $\mathcal{M}_{\mathcal{D},\mathcal{A}}$ includes $\mathsf{H}$ as the sole element of $\mathcal{C}o\mathcal{D}om_{\mathcal{D},\emptyset}$) plus all coatoms of the *upper* attribute slice.

Let us formalize these two itemized points as

**Theorem 5.2.** *For any finite $\mathcal{A}$ and $\mathcal{D}$ such that $|\mathcal{D}| \geq 2$, we have*

$$\mathcal{J}_{\mathcal{D},\mathcal{A}} = \mathfrak{J}(\mathfrak{R}(\mathcal{D}, \mathcal{A})), \qquad \textit{(join-irreducibles)}$$
$$\mathcal{M}_{\mathcal{D},\mathcal{A}} = \mathfrak{M}(\mathfrak{R}(\mathcal{D}, \mathcal{A})). \qquad \textit{(meet-irreducibles)}$$

*Proof.* (join-irreducibles): To prove the $\subseteq$-direction, simply observe that the elements of $\mathcal{J}_{\mathcal{D},\mathcal{A}}$ are exactly the atoms of $\mathfrak{R}(\mathcal{D}, \mathcal{A})$. For the converse, note that

- every element in an $H$-fibre is a join of the atoms of this fibre, as each $H$-fibre has a boolean structure and in the boolean case atomic = atomistic,
- the header elements $(H, \emptyset)$ are joins of elements of $\mathcal{AA}tt_{\mathcal{D},\mathcal{A}}$,
- the atoms of $H$-fibres are joins of header elements with elements of $\mathcal{AA}tt_{\mathcal{D},\mathcal{A}}$.

  Hence, no element of $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ outside $\mathcal{AA}tt_{\mathcal{D},\mathcal{A}}$ can be join-irreducible.

  (meet-irreducibles): This time, the $\supseteq$-direction is easier to show: $\mathcal{M}_{\mathcal{D},\mathcal{A}}$ includes the coatoms of $H$-fibres and of the upper attribute slice. Hence, the basic properties of finite boolean algebras imply all meet-irreducible elements must be contained in $\mathcal{M}_{\mathcal{D},\mathcal{A}}$: every element of $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ can be obtained as an intersection of elements of $\mathcal{M}_{\mathcal{D},\mathcal{A}}$. For the $\subseteq$-direction, it is clear that elements of $\mathcal{C}o\mathcal{A}tt_{\mathcal{D},\mathcal{A}}$ are meet-irreducible, as they are coatoms of the whole $\mathfrak{R}(\mathcal{D}, \mathcal{A})$. This also applies to $\mathsf{H} \in \mathcal{C}o\mathcal{D}om_{\mathcal{D},\emptyset}$. Now take $\mathsf{codom}^H(x) = (H, \mathcal{D}^H - \{x\})$ for a non-empty $H = \{1, \ldots, h\}$ and $x = (x_1, \ldots x_h) \in \mathcal{D}^H$ and assume $\mathsf{codom}^H(x) = r \bowtie s$ for $r, s \neq \mathsf{codom}^H(x)$. That is, $H = H_r \cup H_s$ and

  $$\mathcal{D}^H - \{x\} = \{y \in \mathcal{D}^{H_r \cup H_s} \mid y[H_r] \in B_r \quad \text{and} \quad y[H_s] \in B_s\}.$$

  Note that wlog $H_r \subsetneq H$ and $r \subseteq \mathsf{codom}^{H_r}(z)$ for some $z \in \mathcal{D}^{H_r}$; otherwise, if both $r$ and $s$ were top elements of their respective fibres, their meet would be $(H, \mathcal{D}^H)$. Thus

  $$\mathcal{D}^H - \{x\} \subseteq \{y \in \mathcal{D}^H \mid y[H_r] \neq z\}$$

  and by contraposition

  $$\{y \in \mathcal{D}^H \mid y[H_r] = z\} \subseteq \{x\}. \tag{20}$$

  This means that $z = x[H_r]$. But now take any $i \in H - H_r$, pick any $d \neq x_i$ (here is where we use the assumption that $|\mathcal{D}| \geq 2$) and set

  $$x' := (x_1, \ldots, x_{i-1}, d, x_{i+1}, \ldots, x_h).$$

  Clearly, $x'[H_r] = x[H_r] = z$, contradicting (20). □

**Theorem 5.3.** *Assume $\mathcal{D}, \mathcal{A}$ are finite sets such that $|\mathcal{D}| \geq 2$ and $\mathcal{A} \neq \emptyset$. Then $\mathrm{I}_{\leq}$, $\swarrow$, $\nearrow$ and $\swarrow\!\!\swarrow$ look for $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ as follows:*

| $r =$ <br> $s =$ | $\mathsf{adom}(x)$ <br> $\mathsf{coatt}(a)$ | $\mathsf{aatt}(a)$ <br> $\mathsf{coatt}(b)$ | $\mathsf{adom}(x)$ <br> $\mathsf{codom}^H(y)$ | $\mathsf{aatt}(a)$ <br> $\mathsf{codom}^H(y)$ |
|---|---|---|---|---|
| $r\,\mathrm{I}_{\leq}\,s$ | *always* | $a \neq b$ | $x[H] \neq y$ | $a \notin H$ |
| $r \swarrow s$ | *never* | $a = b$ | $x[H] = y$ | $a \in H$ |
| $r \nearrow s$ | *never* | $a = b$ | $x[H] = y$ | *never* |
| $r \swarrow\!\!\swarrow s$ | *never* | $a = b$ | *always* | *always* |

*Sketch.* For the $\mathrm{I}_{\leq}$-row: this is just spelling out the definition of $\leq$ on $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ as restricted to $\mathcal{J}_{\mathcal{D},\mathcal{A}} \times \mathcal{M}_{\mathcal{D},\mathcal{A}}$.

For the $\swarrow$-row: the set of join-irreducibles consists of only the atoms of the whole lattice, hence $\swarrow$ is just the complement of $\leq$.

This observation already yields $\nearrow \subseteq \swarrow$ and $\swarrow\!\!\!\nearrow = \nearrow$. The last missing piece of information to define $\nearrow$ is provided by the analysis of restriction of $\leq$ to $\mathcal{M}_{\mathcal{D},\mathcal{A}} \times \mathcal{M}_{\mathcal{D},\mathcal{A}}$:

$$\text{for} \quad \begin{array}{ll} r = \mathsf{coatt}(a), & s = \mathsf{coatt}(b), \\ r = \mathsf{coatt}(a), & s = \mathsf{codom}^H(x), \\ r = \mathsf{codom}^H(x), & s = \mathsf{coatt}(a), \\ r = \mathsf{codom}^H(x), & s = \mathsf{codom}^H(y), \end{array} \quad r \leq s \quad \text{iff} \quad \begin{array}{l} \text{never}, \\ \text{never}, \\ a \in H, \\ \text{never}. \end{array}$$

Finally, for $\swarrow\!\!\swarrow$ we need to observe that composing $\swarrow$ with $\nearrow \circ \swarrow$ does not allow reaching any new elements of $\mathcal{C}oAtt_{\mathcal{D},\mathcal{A}}$. As for elements of $\mathcal{M}_{\mathcal{D},\mathcal{A}}$ of the form $\mathsf{codom}^H(y)$, note that

$$\exists h.(h \nearrow \mathsf{coatt}(a) \,\&\, h \swarrow \mathsf{codom}^H(y)) \text{ if } a \in H, \tag{21}$$

$$\exists h.(h \nearrow \mathsf{codom}^{H_x}(x) \,\&\, h \swarrow \mathsf{codom}^{H_y}(y)) \text{ if } x[H_x \cap H_y] = y[H_x \cap H_y]. \tag{22}$$

Furthermore, we have that

- for any $x \in \mathcal{D}^{\mathcal{A}}$ and any $H \subseteq \mathcal{A}$, $\mathsf{adom}(x) \swarrow \mathsf{codom}^H(x[H])$,
- for any $a \in \mathcal{A}$ and any $x \in \mathcal{D}^{\mathcal{A}}$, $\mathsf{aatt}(a) \swarrow \mathsf{codom}^{\mathcal{D}}(x)$.

Using (22), we obtain then that $\mathcal{J}_{\mathcal{D},\mathcal{A}} \times \{\mathsf{H}\} \subseteq \swarrow\!\!\swarrow$ and using (22) again—that $\mathcal{J}_{\mathcal{D},\mathcal{A}} \times \{\mathsf{codom}^H(y)\} \subseteq \swarrow\!\!\swarrow$ for any $y \in \mathcal{D}^{\mathcal{A}}$ and any $H \subseteq \mathcal{A}$. □

**Corollary 5.4.** *If $\mathcal{D}, \mathcal{A}$ are finite sets such that $|\mathcal{D}| \geq 2$ and $\mathcal{A} \neq \emptyset$, then $\mathfrak{R}(\mathcal{D}, \mathcal{A})$ is subdirectly irreducible but not simple.*

*Proof.* Follows immediately from Proposition 5.1 and Theorem 5.3. □

## 6. Extending the Signature and Adding Schema Information

Clearly, it is possible to define more operations on $\mathcal{R}^{\mathsf{H}}_{\text{unr}}$ than those present in $\mathcal{L}_{\mathsf{H}}$. Thus, our first proposal for future study, regardless of the negative result in Corollary 4.8, is a systematic investigation of extensions of the signature. Let us discuss several natural ones; see also [34, 39].

### *6.1. Safe Extensions with Constants and Monotonic Relational Expressions*

We begin with most natural additional constants.

*The top element* $\top := (\emptyset, \{\emptyset\})$. Its inclusion in the signature would be harmless, but at the same time does not appear to improve expressivity in a significant way. Note, however, that if relations with empty header are seen as boolean constants, then $\mathsf{H}$ plays the rôle of $\mathsf{false}$ and $\top$ is necessary to encode $\mathsf{true}$.

*Attribute constants* $\underline{\mathsf{a}} := (\{\mathsf{a}\}, \emptyset)$*, for* $\mathsf{a} \in \mathcal{A}$. Recall again from Sections 1.1 and 1.2 an important difference between our setting and that of both *named SPJR algebra* and *unnamed SPC algebra* in [1, Ch. 4]: in database theory one normally assumes explicit schema information. Our expressions, however, are untyped *query schemes*. On the one hand, $\mathcal{L}_{\mathsf{H}}$ allows, e.g., *projection of* $r$ *to the header of* $s$: $r \oplus (s \bowtie \mathsf{H})$, which does not correspond to any *single* SPJR expression. On the other hand, only with attribute constants we can write the SPJR *projection of* $r$ *to a* *concrete* *header* $\{\mathsf{a}_1, \ldots, \mathsf{a}_n\}$*:* $\pi_{\underline{\mathsf{a}_1, \ldots, \mathsf{a}_n}}(r) := r \oplus \underline{\mathsf{a}_1} \bowtie \ldots \bowtie \underline{\mathsf{a}_n}$. We will return again to the issue of matching the two setups in Section 6.2 below.

*Unary singleton constants* $(\underline{\mathsf{a}} : \mathsf{d}) := (\{\mathsf{a}\}, \{(\mathsf{a} : \mathsf{d})\})$*, for* $\mathsf{a} \in \mathcal{A}$*,* $\mathsf{d} \in \mathcal{D}$. These are among the *base SPJR queries* [1, p. 58]. Note they add more expressivity than attribute constants: whenever the signature includes $(\underline{\mathsf{a}} : \mathsf{d})$ for some $d \in \mathcal{D}$, we have $\underline{\mathsf{a}} = (\underline{\mathsf{a}} : \mathsf{d}) \bowtie \mathsf{H}$. They also allow defining $\top$ as $\top = (\underline{\mathsf{a}} : \mathsf{d}) \oplus \mathsf{H}$ and, more importantly, the SPJR *constant-based selection queries* $\sigma_{\underline{\mathsf{a}}=\mathsf{d}}(r) := r \bowtie (\underline{\mathsf{a}} : \mathsf{d})$.

## 6.2. Equivalence with Monotonic Relational Expressions

As it turns out, the mere addition of unary singleton constants brings our language very close to that of the monotonic relational expressions of Sagiv and Yannakakis [31, Sec. 2.2]. To be more precise, we obtain in this way schema-free (but named!) counterparts of these expressions. The explicit schema-annotation (or header-annotation) discipline necessary to connect these two formalisms is presented in Table 2. It is worth noting that that what database theorists would call schema information, would be called *typing* in the programming community. In the database context, this term would cause obvious confusion as it is usually applied in the setting where *it is impossible for entries from different attributes to be compared* [1, p. 44]. [4] However, the work of Van den Bussche and Waller [9] does in fact use the term *typing* in this context. We highly recommend the reader to compare Table 2 (and the contents of this section in general) with that paper.

## 6.3. Further Extensions Violating Independence Properties

*The bottom element* $\bot := (\mathcal{A}, \emptyset)$. Whenever $\mathcal{A}$ is infinite, including $\bot$ in the signature would exclude subalgebras consisting of relations with finite headers—i.e., exactly those arising from concrete database instances. Another undesirable feature is that the interpretation of $\bot$ depends on $\mathcal{A}$, i.e., the collection of all possible attributes, which is not explicitly supplied by a query expression. In other words, it is domain-independent, but not strictly independent.

*The full relation* $\mathsf{U} := (\mathcal{A}, \mathcal{D}^{\mathcal{A}})$. [39, 34] Its inclusion would destroy even the ordinary domain independence property (d.i.p.). Note that for non-empty $\mathcal{A}$ and $\mathcal{D}$, $\mathsf{U}$ is a complement of $\mathsf{H}$.

*The equality constant* $\mathbf{\Delta} := (\mathcal{A}, \{x \in \mathcal{D}^{\mathcal{A}} \mid \forall a, a'.\, x(a) = x(a')\})$. With it, we can express the *equality-based selection queries*: $\sigma_{\underline{\mathsf{a}}=\underline{\mathsf{b}}}(r) := r \bowtie (\mathbf{\Delta} \oplus \underline{\mathsf{a}} \bowtie \underline{\mathsf{b}})$. Again, the interpretation of $\mathbf{\Delta}$ violates d.i.p.

[4] Another word sometimes used in the context of relations and databases with explicit schema information is *sort(ing)*, but this has obvious potential for confusion as well; we are not concerned with presence or absence of order on either attributes or domain.

Table 2: Equivalence between typed expressions and monotonic relational expressions [31, Sec. 2.2].

***Schema (header) assignment for positive expressions***

$\Sigma$ is a supply of relational symbols $\mathbf{r}$ together with schema (header) information, i.e., $\Sigma = \{\mathbf{r}_1 : H_1, \ldots, \mathbf{r}_n : H_n\}$, where $H \subseteq_{fin} \mathcal{A}$

$$\frac{\mathbf{r} : H \in \Sigma}{\Sigma \vdash \mathbf{r} : H} \qquad \frac{d \in \mathcal{D}, a \in \mathcal{A}}{\Sigma \vdash (\underline{\mathsf{a}} : \mathsf{d}) : \{a\}} \qquad \Sigma \vdash \mathsf{H} : \emptyset$$

$$\frac{\Sigma \vdash r_1 : H_1 \quad \Sigma \vdash r_2 : H_2}{\Sigma \vdash r_1 \bowtie r_2 : H_1 \cup H_2} \qquad \frac{\Sigma \vdash r_1 : H_1 \quad \Sigma \vdash r_2 : H_2}{\Sigma \vdash r_1 \oplus r_2 : H_1 \cap H_2}$$

***Translation $\langle\cdot\rangle$ of monotonic relational expressions [31, Sec. 2.2] into our terms***

Recall $\underline{\mathsf{a}} = (\underline{\mathsf{a}} : \mathsf{d}) \bowtie \mathsf{H}$, for an arbitrary choice of $d \in \mathcal{D}$

$$\langle \pi_{\underline{\mathsf{a_1,\ldots,a_n}}}(r) \rangle = \langle r \rangle \oplus \underline{\mathsf{a_1}} \bowtie \ldots \bowtie \underline{\mathsf{a_n}} \qquad \langle \sigma_{\underline{\mathsf{a}}=\mathsf{d}}(r) \rangle = \langle r \rangle \bowtie (\underline{\mathsf{a}} : \mathsf{d})$$

$$\langle r_1 \bowtie r_2 \rangle = \langle r_1 \rangle \bowtie \langle r_2 \rangle \qquad \langle r_1 \cup r_2 \rangle = \langle r_1 \rangle \oplus \langle r_2 \rangle$$

***Reverse translation $(\cdot)^\Sigma$***

Fix $b \in A$ and distinct $d, e \in \mathcal{D}$. Observe that only for the atomic expressions and $\bowtie$ the translation is independent from the schema information

$$\frac{\mathbf{r} : H \in \Sigma}{(\mathbf{r})^\Sigma = \mathbf{r}} \qquad \frac{d \in \mathcal{D}, a \in \mathcal{A}}{((\underline{\mathsf{a}} : \mathsf{d}))^\Sigma = (\underline{\mathsf{a}} : \mathsf{d})} \qquad (\mathsf{H})^\Sigma = \pi_\emptyset((\underline{\mathsf{b}} : \mathsf{d}) \bowtie (\underline{\mathsf{b}} : \mathsf{e}))$$

$$(r_1 \bowtie r_2)^\Sigma = (r_1)^\Sigma \bowtie (r_2)^\Sigma \qquad \frac{\Sigma \vdash r_1 : \{a_1, \ldots, a_m, b_1, \ldots, b_n\} \quad \Sigma \vdash r_2 : \{b_1, \ldots, b_n, c_1, \ldots, c_k\}}{(r_1 \oplus r_2)^\Sigma = \pi_{\underline{\mathsf{b_1,\ldots,b_n}}}((r_1)^\Sigma) \cup \pi_{\underline{\mathsf{b_1,\ldots,b_n}}}((r_2)^\Sigma)}$$

*6.4. Extensions Respecting Domain Independence*

*The inner equality operator*:

$$[r]^{=} := (H_r, \{x \in \mathcal{D}^{H_r} \mid \exists x' \in r. \exists a' \in H_r. \forall a \in H_r. x(a) = x'(a')\}),$$

It is a domain-independent operation which also allows to define equality-based selection

$$\sigma_{\underline{\mathsf{a}}=\underline{\mathsf{b}}}(r) = r \Join ([r]^{=} \oplus \underline{\mathsf{a}} \Join \underline{\mathsf{b}}).$$

For the purpose of recovering the full setup of Codd's relational algebra, all we need are two additional operations.

The first are standard *attribute renaming* $\rho_{\underline{\mathsf{a}} \mapsto \underline{\mathsf{b}}}(r)$ operators [1, p. 58]. Note that in presence of explicit schema information, attribute renamings can be expressed using a constant which fails d.i.p., namely $[\mathsf{U}]^{=}$. We leave the details out.

The last operation required for expressive completeness is a total version of the difference operator:

*The difference operator* $r - s := (H_r, \{x \in B_r \mid x \notin B_s\})$. This is a very natural extension from the DB point of view [1, Ch. 5], which leads us beyond the SPJRU setting towards the question of *relational completeness* [6]. Here again we break with the partial character of Codd's original operator. Another option would be $(H_{r \cap s}, \{x \in B_r[H_s] \mid x \notin B_s[H_r]\})$, but this one can be defined with the difference operator proposed here as $(r \oplus s) - (s \oplus (r \Join \mathsf{H}))$.

While we do not provide details here, it should be clear how to prove equipollence between the typed version of the formalism with the extensions proposed above and Codd's relational algebra in the spirit of Section 6.2 and Table 2.

## 7. Summary and Future Work

We have seen that relational lattices form an interesting class with rather surprising properties. Unlike Codd's relational algebra, all operations are total and in contrast to the encoding of relational algebras in cylindric algebras, the domain independence property follows automatically. We believe that with the extensions of the language proposed in Section 6, one can ultimately obtain a more natural algebraic treatment of SPRJ(U) operators and relational query languages. Besides, given how well investigated the lattice of varieties of lattices is in general [19], it is intriguing to discover a class of lattices with a natural CS motivation which does not seem to fit anywhere in the existing picture.

We posed a number of questions and problems in the text, in particular Open Problems 3.1, 3.6, 3.7 and 4.10. Without settling them we cannot claim to have grasped how relational lattices behave as an algebraic class. None of them seems trivial, even with the rich supply of algebraic logic tools available in existing literature. Comparison with other settings, like that of Craig [7], Quine [30], other (generalized) algebras of finite sequences and many-sorted cylindric/polyadic algebras [27, Sec. 7.1–7.4 and references therein] and possible attempts at transfer of methods and results would be also of interest.

We would also like to mention the natural question of *representability*:

**Open Problem 7.1** (Hirsch)**.** *Given a finite algebra in the signature $\mathcal{L}_{\mathsf{H}}$ ($\mathcal{L}$), is it decidable whether it belongs to $\mathbb{SP}(\mathcal{R}^{\mathsf{H}}_{\mathrm{unr}})$, $\mathbb{SP}(\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}})$ (resp. $\mathbb{SP}(\mathcal{R}_{\mathrm{unr}})$, $\mathbb{SP}(\mathcal{R}_{\mathrm{fin}})$)?*

We believe that the analysis of the concept structure of finite relational lattices in Section 5 may lead to an algorithm recognizing whether the concept lattice of a given context belongs to $\mathbb{SP}(\mathcal{R}^{\mathsf{H}}_{\mathrm{fin}})$ (or $\mathbb{SP}(\mathcal{R}_{\mathrm{fin}})$). Given the fact that relational lattices have much more meet-irreducible than join-irreducible elements, it is natural to apply the duality promoted by Santocanale and coauthors[32, 11], which is well-tailored for lattices with non-isomorphic sets of meet-irreducible and join-irreducible elements; in fact, efforts in this direction have already been made [32], see Remark 3.8. See also Section 2.3 above for other category-theoretical connections: as suggested therein, the relationship with the work of Abramsky [2] would be of particular interest.

*Acknowledgements.* We would like to thank *Vadim Tropashko* and *Marshall Spight* for introducing the subject to the third author (who in turn introduced it to the other two) and discussing it in the usenet group `comp.databases.theory`, *Maarten Marx*, *Balder ten Cate*, *Jan Paredaens* for additional discussions and general support in an early phase of our cooperation and the *referees* of several incarnations of this work for the comments, which lead to significant improvements in readability of the paper. The first author would also like to acknowledge: a) *Peter Jipsen* for discussions in September 2013 at the Chapman University leading to recovery, rewrite and extension of the material (in particular for Sec. 5) b) Peter Jipsen and *Drew Moshier* for even more discussions and suggestions during their (re-)visit in 2014, c) suggestions by participants of: TACL'09, ALCOP 2010, RAMiCS 2014, the Birmingham TCS seminar and TU Delft Applied Logic Seminar (in particular for Sec. 2.3, 6 and 7), d) *Neil Ghani* for mentioning opfibrations in a conversation, *Alexander Kurz* for making this conversation possible and *Stefan Milius* for an interesting discussion of Abramsky's paper (which sadly was not reflected in the present work).

[1] Abiteboul, S., Hull, R., Vianu, V., 1995. Foundations of Databases. Addison-Wesley.

[2] Abramsky, S., 2013. Relational databases and Bell's theorem. In: Tannen, V., Wong, L., Libkin, L., Fan, W., Tan, W.-C., Fourman, M. (Eds.), In Search of Elegance in the Theory and Practice of Computation. Vol. 8000 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 13–35.
URL `http://dx.doi.org/10.1007/978-3-642-41660-6_2`

[3] Aho, A. V., Sagiv, Y., Ullman, J. D., 1979. Equivalences among relational expressions. SIAM J. Comput. 8 (2), 218–246.
URL `http://dx.doi.org/10.1137/0208017`

[4] Chandra, A. K., Merlin, P. M., 1977. Optimal implementation of conjunctive queries in relational data bases. In: Proceedings of the Ninth Annual ACM Symposium on Theory of Computing. STOC '77. ACM, New York, NY, USA, pp. 77–90.
URL `http://doi.acm.org/10.1145/800105.803397`

[5] Cheng, Q., Gryz, J., Koo, F., Leung, T. Y. C., Liu, L., Qian, X., Schiefer, K. B., 1999. Implementation of two semantic query optimization techniques in db2 universal database. In: Proceedings of the 25th International Conference on Very Large Data Bases. VLDB '99. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 687–698.
URL `http://dl.acm.org/citation.cfm?id=645925.671357`

[6] Codd, E. F., 1970. A relational model of data for large shared data banks. Commun. ACM 13 (6), 377–387.

[7] Craig, W., 1974. Logic in Algebraic Form. Three Languages and Theories. Vol. 72 of Studies in Logic and the Foundations of Mathematics. North Holland.

[8] den Bussche, J. V., Van Gucht, D., Vansummeren, S., 2007. A crash course on database queries. In: PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems. ACM, New York, NY, USA, pp. 143–154.

[9] den Bussche, J. V., Waller, E., 2002. Polymorphic type inference for the relational algebra. Journal of Computer and System Sciences 64 (3), 694 – 718.
URL `http://www.sciencedirect.com/science/article/pii/S0022000001918124`

[10] Düntsch, I., Mikulás, S., 2001. Cylindric structures and dependencies in relational databases. Theor. Comput. Sci. 269 (1-2), 451–468.

[11] Frittella, S., Santocanale, L., 2014. Fixed-point theory in the varieties $\mathcal{D}_n$. In: Höfner, P., Jipsen, P., Kahl, W., Müller, M. E. (Eds.), Relational and Algebraic Methods in Computer Science - 14th International Conference, RAMiCS 2014, Marienstatt, Germany, April 28-May 1, 2014. Proceedings. Vol. 8428 of Lecture Notes in Computer Science. Springer, pp. 446–462.
URL `http://dx.doi.org/10.1007/978-3-319-06251-8_27`

[12] Ganter, B., Wille, R., 1996. Applied lattice theory: Formal concept analysis. In: Grätzer, G. (Ed.), General Lattice Theory, second edition, 2nd Edition. Birkhäuser, pp. 591–606.

[13] Gurevich, Y., 1966. The word problem for certain classes of semigroups. Algebra and Logic 5, 25–35.

[14] Gurevich, Y., Lewis, H. R., 1984. The word problem for cancellation semigroups with zero. The Journal of Symbolic Logic 49 (1), 184–191.
URL `http://www.jstor.org/stable/2274101`

[15] Harrop, R., 1958. On the existence of finite models and decision procedures for propositional calculi. Mathematical Proceedings of the Cambridge Philosophical Society 54, 1–13.
URL `http://journals.cambridge.org/article_S0305004100033120`

[16] Hirsch, R., Hodkinson, I., 2002. Relation Algebras by Games. Vol. 147 of Studies in Logic and the Foundations of Mathematics. Elsevier.

[17] Imieliński, T., Lipski, W., 1984. The relational model of data and cylindric algebras. Journal of Computer and System Sciences 28 (1), 80–102.
URL `http://www.sciencedirect.com/science/article/pii/0022000084900771`

[18] Jacobs, B., 1999. Categorical Logic and Type Theory. No. 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam.

[19] Jipsen, P., Rose, H., 1992. Varieties of Lattices. Vol. 1533 of Lecture Notes in Mathematics. Springer Verlag.

[20] Jipsen, P., Rose, H., 1998. Varieties of lattices. In: Grätzer, G. (Ed.), General Lattice Theory. Birkhäuser Verlag, pp. 555–574, appendix F to the second edition.

[21] Johnson, D. S., Klug, A., 1982. Testing containment of conjunctive queries under functional and inclusion dependencies.

In: Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. PODS '82. ACM, New York, NY, USA, pp. 164–169.
URL http://doi.acm.org/10.1145/588111.588138

[22] Johnson, M., Rosebrugh, R., Dec. 2007. Fibrations and universal view updatability. Theoretical Computer Science 388 (1–3), 109–129.
URL http://dx.doi.org/10.1016/j.tcs.2007.06.004

[23] Johnson, M., Rosebrugh, R., Wood, R. J., Feb. 2012. Lenses, fibrations and universal translations. Mathematical Structures in Computer Science 22 (1), 25–42.
URL http://dx.doi.org/10.1017/S0960129511000442

[24] Litak, T., Mikulás, S., Hidders, J., 2014. Relational lattices. In: Höfner, P., Jipsen, P., Kahl, W., Müller, M. E. (Eds.), Relational and Algebraic Methods in Computer Science 2014 (RAMiCS). Vol. 8428 of Lecture Notes in Computer Science. Springer International Publishing, pp. 327–343.
URL http://dx.doi.org/10.1007/978-3-319-06251-8_20

[25] Maddux, R., 1980. The equational theory of $CA_3$ is undecidable. The Journal of Symbolic Logic 45 (2), 311–316.
URL http://www.jstor.org/stable/2273191

[26] McKenzie, R., 1972. Equational bases and non-modular lattice varieties. Trans. Amer. Math. Soc. 174, 1–43.

[27] Németi, I., 1991. Algebraization of quantifier logics, an introductory overview. Studia Logica: An International Journal for Symbolic Logic 50 (3/4), pp. 485–569.
URL http://www.jstor.org/stable/20015599

[28] Padmanabhan, R., McCune, W., Veroff, R., 2007. Lattice laws forcing distributivity under unique complementation. Houston Journal of Mathematics 33 (2), 391–401.

[29] Post, E. L., 1947. Recursive unsolvability of a problem of Thue. The Journal of Symbolic Logic 12 (1), 1–11.
URL http://www.jstor.org/stable/2267170

[30] Quine, W. V., 1936. Toward a calculus of concepts. The Journal of Symbolic Logic 1 (1), pp. 2–25.
URL http://www.jstor.org/stable/2269323

[31] Sagiv, Y., Yannakakis, M., Oct. 1980. Equivalences among relational expressions with the union and difference operators. J. ACM 27 (4), 633–655.
URL http://doi.acm.org/10.1145/322217.322221

[32] Santocanale, L., Nov. 2009. A duality for finite lattices.
URL https://hal.archives-ouvertes.fr/hal-00432113

[33] Santocanale, L., 2015. Relational lattices via duality, conference talk at TACL.

[34] Spight, M., Tropashko, V., 2006. First steps in relational lattice. http://arxiv.org/abs/cs/0603044.
URL http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0603044

[35] Spivak, D. I., 12 2014. Database queries and constraints via lifting problems. Mathematical Structures in Computer Science 24.
URL http://journals.cambridge.org/article_S0960129513000479

[36] Stanley, R. P., 1972. Supersolvable lattices. Algebra Universalis 2 (1), 197–217.
URL http://dx.doi.org/10.1007/BF02945028

[37] Stern, M., 1999. Semimodular Lattices. Vol. 73 of Encyclopedia of Mathematics and its Applications. Cambridge University Press.

[38] Tropashko, V., 2005. Relational algebra as non-distributive lattice. http://arxiv.org/abs/cs/0501053.
URL http://arxiv.org/abs/cs/0501053

[39] Tropashko, V., 2009–13. The website of QBQL: Prototype of relational lattice system, https://code.google.com/p/qbql/.

[40] Vardi, M. Y., 1981. The decision problem for database dependencies. Inf. Process. Lett. 12 (5), 251–254.
URL http://dx.doi.org/10.1016/0020-0190(81)90025-9

## Appendix A. Theorem 3.4, Clause 7:

```
formulas(assumptions).
x ^ y = y ^ x.
(x ^ y) ^ z = x ^ (y ^ z).
x v y = y v x.
(x v y) v z = x v (y v z).
x v (x ^ y) = x.
x ^ (x v y) = x.
UpMe(x,y,z) = x ^ (y v z).
LoMe(x,y,z) = (x ^ y) v (x ^ z).
UpJo(x,y,z) = (x v y) ^ (x v z).
LoJo(x,y,z) = x v (y ^ z).
end_of_list.
```

**formulas(goals).**
```
(all x1 all y1 all w UpMe(a ^ x1,y1,w) v (y1 ^ w) = (((a ^ x1) ^ y1) v w) ^ (((a ^ x1) ^
    w) v y1)) ->

UpMe(a ^ z1,z2,z3) = LoMe(a ^ z1,z2,z3).
```
**end_of_list.**

.......

============================== **PROOF** =================================

% **Proof** 1 at 45.03 (+ 0.26) seconds.
% **Length of proof** is 66.
% **Level of proof** is 14.
% Maximum clause weight is 35.
% Given clauses 464.

```
1 (all x1 all y1 all w UpMe(a ^ x1,y1,w) v (y1 ^ w) = (((a ^ x1) ^ y1) v w) ^ (((a ^ x1)
    ^ w) v y1)) -> UpMe(a ^ z1,z2,z3) = LoMe(a ^ z1,z2,z3) # label(non_clause) # label(goal).
    [goal].
2 x ^ y = y ^ x. [assumption].
3 (x ^ y) ^ z = x ^ (y ^ z). [assumption].
4 x v y = y v x. [assumption].
5 (x v y) v z = x v (y v z). [assumption].
6 x v (x ^ y) = x. [assumption].
7 x ^ (x v y) = x. [assumption].
8 UpMe(x,y,z) = x ^ (y v z). [assumption].
9 LoMe(x,y,z) = (x ^ y) v (x ^ z). [assumption].
12 UpMe(a ^ x,y,z) v (y ^ z) = (((a ^ x) ^ y) v z) ^ (((a ^ x) ^ z) v y). [deny(1)].
13 (x ^ y) v (a ^ (z ^ (x v y))) = ((a ^ (z ^ x)) v y) ^ ((a ^ (z ^ y)) v x). [copy(12),
    rewrite([8(3),3(4),4(6),3(9),3(13)])].
14 LoMe(a ^ c1,c2,c3) != UpMe(a ^ c1,c2,c3). [deny(1)].
15 (c2 ^ (a ^ c1)) v (c3 ^ (a ^ c1)) != a ^ (c1 ^ (c2 v c3)). [copy(14),rewrite([9(6),2(5)
    ,2(10),8(17),3(18)])].
16 x ^ (y ^ z) = z ^ (x ^ y). [para(3(a,1),2(a,1))].
17 x ^ (y ^ z) = y ^ (x ^ z). [para(2(a,1),3(a,1,1)),rewrite([3(2)])].
18 (a ^ (c1 ^ c2)) v (a ^ (c1 ^ c3)) != a ^ (c1 ^ (c2 v c3)). [back_rewrite(15),rewrite([16(5)
    ,2(4),17(5),16(10),2(9),17(10)])].
20 x v (y v z) = y v (x v z). [para(4(a,1),5(a,1,1)),rewrite([5(2)])].
21 x v (y ^ x) = x. [para(2(a,1),6(a,1,2))].
22 (x ^ y) v (x ^ (y ^ z)) = x ^ y. [para(3(a,1),6(a,1,2))].
23 x v ((x ^ y) v z) = x v z. [para(6(a,1),5(a,1,1)),flip(a)].
26 x ^ (y ^ ((x ^ y) v z)) = x ^ y. [para(7(a,1),3(a,1)),flip(a)].
27 x ^ (y v x) = x. [para(4(a,1),7(a,1,2))].
28 (x v y) ^ (x v (y v z)) = x v y. [para(5(a,1),7(a,1,2))].
29 x v x = x. [para(7(a,1),6(a,1,2))].
30 x ^ x = x. [para(6(a,1),7(a,1,2))].
38 (x ^ (y ^ z)) v (a ^ (u ^ ((x ^ y) v z))) = ((a ^ (u ^ (x ^ y))) v z) ^ ((a ^ (u ^ z))
     v (x ^ y)). [para(3(a,1),13(a,1,1))].
63 x ^ (y ^ (x v z)) = y ^ x. [para(7(a,1),17(a,1,2)),flip(a)].
```

76 x v (y v x) = y v x. *[para(29(a,1),5(a,2,2)),rewrite([4(2)])]*.
79 x ^ (x ^ y) = x ^ y. *[para(30(a,1),3(a,1,1)),flip(a)]*.
81 x ^ (y ^ x) = y ^ x. *[para(30(a,1),3(a,2,2)),rewrite([2(2)])]*.
82 (x ^ y) v (a ^ (x v y)) = ((a ^ x) v y) ^ ((a ^ y) v x). *[para(30(a,1),13(a,1,2,2)),rewrite([2(8),7(8),2(11),27(11)])]*.
88 x v (y ^ (z ^ x)) = x. *[para(3(a,1),21(a,1,2))]*.
89 x v ((y ^ x) v z) = x v z. *[para(21(a,1),5(a,1,1)),flip(a)]*.
95 x ^ ((y v x) ^ z) = x ^ z. *[para(27(a,1),3(a,1,1)),flip(a)]*.
97 x ^ (y v (z v x)) = x. *[para(5(a,1),27(a,1,2))]*.
111 x v (y v (z ^ x)) = y v x. *[para(21(a,1),20(a,1,2)),flip(a)]*.
132 x v (y v (z v x)) = y v (z v x). *[para(5(a,1),76(a,1,2)),rewrite([5(5)])]*.
141 (x ^ y) v ((x ^ (y ^ z)) v u) = (x ^ y) v u. *[para(22(a,1),5(a,1,1)),flip(a)]*.
150 x ^ (y ^ (z ^ x)) = y ^ (z ^ x). *[para(3(a,1),81(a,1,2)),rewrite([3(5)])]*.
155 (x ^ y) v (y ^ x) = x ^ y. *[para(81(a,1),22(a,1,2))]*.
158 x v ((y ^ (z ^ x)) v u) = x v u. *[para(88(a,1),5(a,1,1)),flip(a)]*.
176 a ^ (x ^ ((y v z) ^ (u v (((a ^ (x ^ y)) v z) ^ ((a ^ (x ^ z)) v y))))) = a ^ (x ^ (y v z)). *[para(13(a,1),97(a,1,2,2)),rewrite([3(15),3(14)])]*.
195 (x v y) ^ ((x ^ z) v y) = (x ^ z) v y. *[para(23(a,1),27(a,1,2)),rewrite([2(4)])]*.
197 (x ^ y) v (z v x) = z v x. *[para(76(a,1),23(a,2)),rewrite([132(4)])]*.
220 (x ^ y) v (x ^ (y v z)) = x ^ (y v z). *[para(63(a,1),21(a,1,2)),rewrite([4(4)])]*.
228 (x v y) ^ ((z ^ x) v y) = (z ^ x) v y. *[para(89(a,1),27(a,1,2)),rewrite([2(4)])]*.
244 (x v y) ^ (z ^ y) = z ^ y. *[para(81(a,1),95(a,2)),rewrite([150(4)])]*.
321 (a ^ x) v (((a ^ x) v y) ^ z) = ((a ^ x) v z) ^ ((a ^ x) v y). *[para(26(a,1),13(a,2,1,1)),rewrite([5(9),26(11),4(7),20(17),141(17)])]*.
336 (x ^ y) v (z v (y ^ x)) = z v (y ^ x). *[para(155(a,1),5(a,2,2)),rewrite([4(4)])]*.
341 (x ^ (y ^ z)) v (y ^ x) = x ^ y. *[para(155(a,1),23(a,2)),rewrite([3(3),336(6)])]*.
398 (x v y) ^ (y v x) = x v y. *[para(76(a,1),28(a,1,2))]*.
565 (x v (y v z)) ^ (z v y) = z v y. *[para(398(a,1),244(a,1,2)),rewrite([398(7)])]*.
1281 (x v y) ^ ((z ^ (u ^ x)) v y) = (z ^ (u ^ x)) v y. *[para(158(a,1),27(a,1,2)),rewrite([2(5)])]*.
3918 (x v y) ^ ((z ^ y) v x) = (z ^ y) v x. *[para(111(a,1),565(a,1,1))]*.
6421 x ^ ((x ^ y) v (x ^ z)) = (x ^ y) v (x ^ z). *[para(6(a,1),195(a,1,1))]*.
6979 (x ^ (y ^ z)) v (x ^ (u v y)) = x ^ (u v y). *[para(197(a,1),220(a,1,2,2)),rewrite([197(8)])]*.
7074 x ^ ((y ^ z) v (y ^ x)) = x ^ y. *[para(341(a,1),228(a,1,2)),rewrite([2(5),3(5),6421(4),341(8)])]*.
7542 x ^ ((y ^ z) v (x ^ y)) = x ^ y. *[para(2(a,1),7074(a,1,2,2))]*.
7744 x ^ ((y ^ z) v (x ^ (u v z))) = x ^ (u v z). *[para(244(a,1),7542(a,1,2,1))]*.
8021 (a ^ x) v (a ^ y) = a ^ (x v y). *[para(82(a,2),38(a,1,2,2)),rewrite([7744(10),6979(7),2(10),3(10),27(9),79(7),2(11),26(12),3918(12)]),flip(a)]*.
8518 a ^ ((c1 ^ c2) v (c1 ^ c3)) != a ^ (c1 ^ (c2 v c3)). *[back_rewrite(18),rewrite([8021(11)])]*.
9294 a ^ ((a ^ x) v y) = a ^ (x v y). *[para(79(a,1),8021(a,1,1)),rewrite([8021(5)]),flip(a)]*.
10727 a ^ (x ^ ((a ^ y) v z)) = x ^ (a ^ (y v z)). *[para(9294(a,1),17(a,1,2)),flip(a)]*.
15426 x ^ (a ^ ((x ^ y) v z)) = a ^ (x ^ (y v z)). *[para(6(a,1),176(a,1,2,2,2)),rewrite([1281(7),10727(7)])]*.
36559 a ^ ((x ^ y) v (x ^ z)) = a ^ (x ^ (y v z)). *[para(341(a,1),321(a,1,2,1)),rewrite([3(6),8021(7),341(15),2(12),3(12),10727(12),15426(10)])]*.
36560 $F. *[resolve(36559,a,8518,a)]*.

============================== **end of proof** ==========================

============================== **STATISTICS** ============================

```
Given=464. Generated=384816. Kept=36556. proofs=1.
Usable=432. Sos=19999. Demods=18086. Limbo=2, Disabled=16134. Hints=0.
Kept_by_rule=0, Deleted_by_rule=385.
Forward_subsumed=290602. Back_subsumed=991.
Sos_limit_deleted=57273. Sos_displaced=7640. Sos_removed=0.
New_demodulators=28671 (6 lex), Back_demodulated=7489. Back_unit_deleted=0.
Demod_attempts=7921128. Demod_rewrites=1153227.
Res_instance_prunes=0. Para_instance_prunes=0. Basic_paramod_prunes=0.
Nonunit_fsub_feature_tests=0. Nonunit_bsub_feature_tests=0.
Megabytes=75.19.
User_CPU=45.03, System_CPU=0.26, Wall_clock=46.
```

============================== **end of statistics** =====================

============================== **end of search** =========================

**THEOREM PROVED**

**Appendix B. Theorem 3.4, Clause 8:**

**formulas(assumptions)**.

```
x ^ y = y ^ x.
(x ^ y) ^ z = x ^ (y ^ z).
x v y = y v x.
(x v y) v z = x v (y v z).
x v (x ^ y) = x.
x ^ (x v y) = x.
UpMe(x,y,z) = x ^ (y v z).
LoMe(x,y,z) = (x ^ y) v (x ^ z).
UpJo(x,y,z) = (x v y) ^ (x v z).
LoJo(x,y,z) = x v (y ^ z).
UpMe(a ^ x1,y1,z1) v (y1 ^ z1) = (((a ^ x1) ^ y1) v z1) ^ (((a ^ x1) ^ z1) v y1).
UpMe(x,y,z) = UpMe(x,y,a ^ z) v UpMe(x,z,a ^ y).
```

**end_of_list**.

**formulas(goals)**.

```
(all x2 all y2 all z2 (UpMe(a,x2,y2) = UpMe(a,x2,z2) -> UpMe(x2,y2,z2) = LoMe(x2,y2,z2)))
    .
```

**end_of_list**.

....

============================== **PROOF** ================================

```
\% Proof 1 at 222.55 (+ 1.51) seconds.
\% Length of proof is 195.
\% Level of proof is 24.
\% Maximum clause weight is 47.
\% Given clauses 1611.

1 (all x2 all y2 all z2 (UpMe(a,x2,y2) = UpMe(a,x2,z2) -> UpMe(x2,y2,z2) = LoMe(x2,y2,z2)
    )) # label(non_clause) # label(goal). [goal].
2 x ^ y = y ^ x. [assumption].
3 (x ^ y) ^ z = x ^ (y ^ z). [assumption].
4 x v y = y v x. [assumption].
5 (x v y) v z = x v (y v z). [assumption].
6 x v (x ^ y) = x. [assumption].
7 x ^ (x v y) = x. [assumption].
8 UpMe(x,y,z) = x ^ (y v z). [assumption].
9 LoMe(x,y,z) = (x ^ y) v (x ^ z). [assumption].
12 UpMe(a ^ x,y,z) v (y ^ z) = (((a ^ x) ^ y) v z) ^ (((a ^ x) ^ z) v y). [assumption].
13 (x ^ y) v (a ^ (z ^ (x v y))) = ((a ^ (z ^ x)) v y) ^ ((a ^ (z ^ y)) v x). [copy(12),
    rewrite([8(3),3(4),4(6),3(9),3(13)])].
14 UpMe(x,y,z) = UpMe(x,y,a ^ z) v UpMe(x,z,a ^ y). [assumption].
15 (x ^ (y v (a ^ z))) v (x ^ (z v (a ^ y))) = x ^ (y v z). [copy(14),rewrite([8(1),8(5),8(9)]),
    flip(a)].
16 UpMe(a,c1,c3) = UpMe(a,c1,c2). [deny(1)].
17 a ^ (c1 v c3) = a ^ (c1 v c2). [copy(16),rewrite([8(4),8(9)])].
18 LoMe(c1,c2,c3) != UpMe(c1,c2,c3). [deny(1)].
19 (c1 ^ c2) v (c1 ^ c3) != c1 ^ (c2 v c3). [copy(18),rewrite([9(4),8(11)])].
21 x ^ (y ^ z) = y ^ (x ^ z). [para(2(a,1),3(a,1,1)),rewrite([3(2)])].
23 x v (y v z) = y v (x v z). [para(4(a,1),5(a,1,1)),rewrite([5(2)])].
24 x v (y ^ x) = x. [para(2(a,1),6(a,1,2))].
25 (x ^ y) v (x ^ (y ^ z)) = x ^ y. [para(3(a,1),6(a,1,2))].
26 x v ((x ^ y) v z) = x v z. [para(6(a,1),5(a,1,1)),flip(a)].
27 x v (y v ((x v y) ^ z)) = x v y. [para(6(a,1),5(a,1)),flip(a)].
28 x ^ ((x v y) ^ z) = x ^ z. [para(7(a,1),3(a,1,1)),flip(a)].
29 x ^ (y ^ ((x ^ y) v z)) = x ^ y. [para(7(a,1),3(a,1)),flip(a)].
30 x ^ (y v x) = x. [para(4(a,1),7(a,1,2))].
31 (x v y) ^ (x v (y v z)) = x v y. [para(5(a,1),7(a,1,2))].
32 x v x = x. [para(7(a,1),6(a,1,2))].
33 x ^ x = x. [para(6(a,1),7(a,1,2))].
36 (x ^ y) v (a ^ ((x v y) ^ z)) = ((a ^ (z ^ x)) v y) ^ ((a ^ (z ^ y)) v x). [para(2(a,1)
    ,13(a,1,2,2))].
37 (x ^ y) v (a ^ (z ^ (x v y))) = ((a ^ (x ^ z)) v y) ^ ((a ^ (z ^ y)) v x). [para(2(a,1)
    ,13(a,2,1,1,2))].
41 (x ^ (y ^ z)) v (a ^ (u ^ ((x ^ y) v z))) = ((a ^ (u ^ (x ^ y))) v z) ^ ((a ^ (u ^ z))
     v (x ^ y)). [para(3(a,1),13(a,1,1))].
44 (x ^ y) v (a ^ (z ^ (x v y))) = (y v (a ^ (z ^ x))) ^ ((a ^ (z ^ y)) v x). [para(4(a,1)
    ,13(a,2,1))].
55 ((a ^ x) v y) ^ (x v ((a ^ (x ^ y)) v z)) = (a ^ x) v ((x v z) ^ y). [para(7(a,1),13(a
    ,2,1,1,2)),rewrite([5(5),7(6),4(5),23(13)]),flip(a)].
58 (((a ^ (x ^ y)) v z) ^ u) v (((a ^ (x ^ y)) v z) ^ ((a ^ (x ^ z)) v y)) = (y ^ z) v ((
    a ^ (x ^ (y v z))) v (((a ^ (x ^ y)) v z) ^ u)). [para(13(a,2),9(a,2,2)),rewrite([9(9),23(27)
    ,4(26)])].
```

68 (x ^ (y v (a ^ z))) v ((z v (a ^ y)) ^ x) = x ^ (y v z). *[para(2(a,1),15(a,1,2))].*

83 a ^ ((c1 v c3) ^ x) = a ^ ((c1 v c2) ^ x). *[para(17(a,1),3(a,1,1)),rewrite([3(6)]),flip(a)].*

90 x v (y ^ (x ^ z)) = x. *[para(21(a,1),6(a,1,2))].*

91 x ^ (y ^ (x v z)) = y ^ x. *[para(7(a,1),21(a,1,2)),flip(a)].*

92 (x ^ (y ^ z)) v (a ^ (u ^ (y v (x ^ z)))) = ((a ^ (u ^ y)) v (x ^ z)) ^ ((a ^ (u ^ (x ^ z))) v y). *[para(21(a,1),13(a,1,1))].*

108 x v (y v x) = y v x. *[para(32(a,1),5(a,2,2)),rewrite([4(2)])].*

112 x ^ (x ^ y) = x ^ y. *[para(33(a,1),3(a,1,1)),flip(a)].*

114 x ^ (y ^ x) = y ^ x. *[para(33(a,1),3(a,2,2)),rewrite([2(2)])].*

115 (x ^ y) v (a ^ (x v y)) = ((a ^ x) v y) ^ ((a ^ y) v x). *[para(33(a,1),13(a,1,2,2)),rewrite( [2(8),7(8),2(11),30(11)])].*

132 x v (y ^ (z ^ x)) = x. *[para(3(a,1),24(a,1,2))].*

133 x v ((y ^ x) v z) = x v z. *[para(24(a,1),5(a,1,1)),flip(a)].*

140 (x ^ y) v (x ^ (z ^ y)) = x ^ y. *[para(21(a,1),24(a,1,2))].*

141 x ^ ((y v x) ^ z) = x ^ z. *[para(30(a,1),3(a,1,1)),flip(a)].*

142 x ^ (y ^ (z v (x ^ y))) = x ^ y. *[para(30(a,1),3(a,1)),flip(a)].*

143 x ^ (y v (z v x)) = x. *[para(5(a,1),30(a,1,2))].*

146 ((a ^ x) v y) ^ ((a ^ (x ^ y)) v (z v x)) = (a ^ x) v (y ^ (z v x)). *[para(30(a,1),13(a ,2,2,1,2)),rewrite([143(6),4(5),2(14)]),flip(a)].*

149 x ^ (y ^ (z v x)) = y ^ x. *[para(30(a,1),21(a,1,2)),flip(a)].*

152 x v (y v (x ^ z)) = y v x. *[para(6(a,1),23(a,1,2)),flip(a)].*

153 x ^ (y v (x v z)) = x. *[para(23(a,1),7(a,1,2))].*

159 x v (y v (z ^ x)) = y v x. *[para(24(a,1),23(a,1,2)),flip(a)].*

160 (x v y) ^ (x v (z v y)) = x v y. *[para(23(a,1),30(a,1,2))].*

163 x v ((y ^ (x ^ z)) v u) = x v u. *[para(90(a,1),5(a,1,1)),flip(a)].*

170 x ^ (y ^ (x ^ z)) = y ^ (x ^ z). *[para(90(a,1),30(a,1,2)),rewrite([2(3)])].*

171 x v (y v (z ^ (x ^ u))) = y v x. *[para(90(a,1),23(a,1,2)),flip(a)].*

174 (a ^ x) v ((x v y) ^ z) = ((a ^ x) v z) ^ (x v y). *[back_rewrite(55),rewrite([163(8)]),flip(a) ].*

185 x v (y v (z v x)) = y v (z v x). *[para(5(a,1),108(a,1,2)),rewrite([5(5)])].*

196 (x ^ y) v ((x ^ (y ^ z)) v u) = (x ^ y) v u. *[para(25(a,1),5(a,1,1)),flip(a)].*

207 x ^ (y ^ (z ^ x)) = y ^ (z ^ x). *[para(3(a,1),114(a,1,2)),rewrite([3(5)])].*

213 (x ^ y) v (y ^ x) = x ^ y. *[para(114(a,1),25(a,1,2))].*

215 x v (y ^ (z ^ (u ^ x))) = x. *[para(3(a,1),132(a,1,2,2))].*

216 x v ((y ^ (z ^ x)) v u) = x v u. *[para(132(a,1),5(a,1,1)),flip(a)].*

217 x v (y v (z ^ (u ^ (x v y)))) = x v y. *[para(132(a,1),5(a,1)),flip(a)].*

231 x ^ (y ^ (z v (u v (x ^ y)))) = x ^ y. *[para(143(a,1),3(a,1)),flip(a)].*

232 x ^ (y v (z v (u v x))) = x. *[para(5(a,1),143(a,1,2,2))].*

236 a ^ (x ^ ((y v z) ^ (u v (((a ^ (x ^ y)) v z) ^ ((a ^ (x ^ z)) v y))))) = a ^ (x ^ (y v z)). *[para(13(a,1),143(a,1,2,2)),rewrite([3(15),3(14)])].*

259 x v (((a ^ (y ^ x)) v z) ^ ((a ^ (y ^ z)) v x)) = x v (a ^ (y ^ (x v z))). *[para(13(a ,1),26(a,1,2))].*

261 (x v y) ^ ((x ^ z) v y) = (x ^ z) v y. *[para(26(a,1),30(a,1,2)),rewrite([2(4)])].*

262 x v (y v ((x ^ z) v u)) = y v (x v u). *[para(26(a,1),23(a,1,2)),flip(a)].*

263 (x ^ y) v (z v x) = z v x. *[para(108(a,1),26(a,2)),rewrite([185(4)])].*

267 (x ^ y) v (a ^ (z ^ (x v ((x v u) ^ y)))) = (x v (a ^ (z ^ ((x v u) ^ y)))) ^ ((a ^ ( z ^ x)) v ((x v u) ^ y)). *[para(28(a,1),13(a,1,1)),rewrite([4(20),2(21)])].*

276 (x ^ y) v ((x v z) ^ y) = (x v z) ^ y. *[para(28(a,1),24(a,1,2)),rewrite([4(4)])].*

277 (x v y) ^ (z ^ x) = z ^ x. *[para(114(a,1),28(a,2)),rewrite([207(4)])].*

280 x ^ (y ^ (z ^ (x v u))) = y ^ (z ^ x). *[para(3(a,1),91(a,1,2)),rewrite([3(6)])].*

292 (x ^ y) v (x ^ (y v z)) = x ^ (y v z). *[para(91(a,1),24(a,1,2)),rewrite([4(4)])].*

297 x v (((a ^ (y ^ z)) v x) ^ ((a ^ (y ^ x)) v z)) = x v (a ^ (y ^ (z v x))). *[para(13(a,1),133(a,1,2))]*.

301 (x ^ y) v ((x ^ (z ^ y)) v u) = (x ^ y) v u. *[para(21(a,1),133(a,1,2,1))]*.

302 (x v y) ^ ((z ^ x) v y) = (z ^ x) v y. *[para(133(a,1),30(a,1,2)),rewrite([2(4)])]*.

304 (x ^ y) v (z v y) = z v y. *[para(108(a,1),133(a,2)),rewrite([185(4)])]*.

321 (x v y) ^ (z ^ y) = z ^ y. *[para(114(a,1),141(a,2)),rewrite([207(4)])]*.

338 x v (y v (((x ^ z) v y) ^ u)) = x v y. *[para(27(a,1),26(a,1,2)),rewrite([26(3)]),flip(a)]*.

352 a ^ (c3 ^ (c1 v c2)) = a ^ c3. *[para(17(a,1),149(a,1,2)),rewrite([21(7)])]*.

353 (x ^ y) v (x ^ (z v y)) = x ^ (z v y). *[para(149(a,1),24(a,1,2)),rewrite([4(4)])]*.

380 (x v y) ^ (x v (z ^ y)) = x v (z ^ y). *[para(159(a,1),30(a,1,2)),rewrite([2(4)])]*.

403 x ^ (y ^ ((y ^ x) v z)) = x ^ y. *[para(2(a,1),29(a,1,2,2,1))]*.

407 (a ^ x) v (((a ^ x) v y) ^ z) = ((a ^ x) v z) ^ ((a ^ x) v y). *[para(29(a,1),13(a,2,1,1)),rewrite([5(9),29(11),4(7),23(17),196(17)])]*.

424 (x ^ y) v (z v (y ^ x)) = z v (y ^ x). *[para(213(a,1),5(a,2,2)),rewrite([4(4)])]*.

428 x ^ (y ^ (z v (y ^ x))) = x ^ y. *[para(213(a,1),143(a,1,2,2)),rewrite([3(4)])]*.

429 (x ^ (y ^ z)) v (y ^ x) = x ^ y. *[para(213(a,1),26(a,2)),rewrite([3(3),424(6)])]*.

431 x v ((y ^ (z ^ (u ^ x))) v w) = x v w. *[para(215(a,1),5(a,1,1)),flip(a)]*.

458 a ^ (x ^ ((y v z) ^ (u v (w v (((a ^ (x ^ y)) v z) ^ ((a ^ (x ^ z)) v y)))))) = a ^ (x ^ (y v z)). *[para(13(a,1),232(a,1,2,2,2)),rewrite([3(16),3(15)])]*.

461 x ^ ((y v (a ^ z)) ^ (u v (w v (x ^ (z v y))))) = x ^ (y v (a ^ z)). *[para(15(a,1),232(a,1,2,2,2)),rewrite([3(9)])]*.

483 (x v y) ^ (y v (x v z)) = y v x. *[para(4(a,1),31(a,1,1))]*.

493 (x v y) ^ (y v x) = x v y. *[para(108(a,1),31(a,1,2))]*.

495 (x ^ (y ^ z)) v (u v (x ^ y)) = u v (x ^ y). *[para(3(a,1),263(a,1,1))]*.

503 (x ^ (y ^ z)) v (u v y) = u v y. *[para(21(a,1),263(a,1,1))]*.

512 (a ^ x) v (y ^ (z v x)) = ((a ^ x) v y) ^ (z v x). *[back_rewrite(146),rewrite([503(8)]),flip(a)]*.

514 (x v y) ^ (z ^ (u ^ x)) = z ^ (u ^ x). *[para(3(a,1),277(a,1,2)),rewrite([3(6)])]*.

516 (x ^ y) v (a ^ (z ^ (u v y))) = (y v u) ^ ((a ^ (z ^ (y v u))) v (x ^ y)). *[para(277(a,1),13(a,1,1)),rewrite([5(5),159(5),23(18),431(18),2(14)])]*.

534 (x ^ (y ^ z)) v (u v z) = u v z. *[para(3(a,1),304(a,1,1))]*.

566 ((a ^ (x ^ y)) v z) ^ (((a ^ (x ^ z)) v y) ^ (u ^ (a ^ (x ^ (y v z))))) = u ^ (a ^ (x ^ (y v z))). *[para(13(a,1),321(a,1,1)),rewrite([3(15)])]*.

582 (x v y) ^ (z ^ (x v (u ^ y))) = z ^ (x v (u ^ y)). *[para(159(a,1),321(a,1,1))]*.

674 c1 v (c2 v (a ^ c3)) = c1 v c2. *[para(352(a,1),132(a,1,2)),rewrite([4(7),23(7),4(6)])]*.

695 x v (y v (z ^ (y v x))) = x v y. *[para(493(a,1),132(a,1,2,2)),rewrite([5(4)])]*.

698 (x v (y v z)) ^ (z v y) = z v y. *[para(493(a,1),321(a,1,2)),rewrite([493(7)])]*.

712 a ^ (c3 ^ (x v (c1 v c2))) = a ^ c3. *[para(674(a,1),232(a,1,2,2)),rewrite([3(8)])]*.

847 x ^ (y ^ (z ^ (u v (x ^ z)))) = x ^ (y ^ z). *[para(140(a,1),143(a,1,2,2)),rewrite([3(5),3(4)])]*.

851 (x ^ (y ^ z)) v (x ^ (y ^ (z v u))) = x ^ (y ^ (z v u)). *[para(91(a,1),140(a,1,2,2)),rewrite([4(6)])]*.

870 (a ^ x) v (y ^ (z v (a ^ x))) = (z v (a ^ x)) ^ ((a ^ x) v y). *[para(142(a,1),13(a,2,2,1)),rewrite([231(11),4(7),495(14)])]*.

904 (x v y) ^ (x v (z ^ (u ^ y))) = x v (z ^ (u ^ y)). *[para(132(a,1),160(a,1,2,2)),rewrite([2(5)])]*.

1373 (x ^ y) v z = z v (((a ^ (x ^ z)) v y) ^ ((a ^ (z ^ y)) v x)). *[para(37(a,1),171(a,1,2)),flip(a)]*.

1688 (x v y) ^ ((z ^ (u ^ x)) v y) = (z ^ (u ^ x)) v y. *[para(216(a,1),30(a,1,2)),rewrite([2(5)])]*.

2786 (a ^ x) v (y ^ (z ^ (x v u))) = (x v u) ^ ((a ^ x) v (y ^ z)). *[para(7(a,1),41(a,2,2,1,2)),rewrite([153(8),4(6),23(12),163(12)])]*.

4300 (x ^ y) v (a ^ (z ^ (x v y))) = (y v (a ^ (z ^ x))) ^ (x v (a ^ (z ^ y))). *[para(4(a,1),44(a,2,2))]*.

4305 (x ^ y) v (a ^ x) = x ^ (y v (a ^ x)). *[para(7(a,1),44(a,1,2,2)),rewrite([33(6),4(11),90(11),2(8)])]*.

4488 x ^ ((a ^ y) v (a ^ x)) = a ^ x. *[para(403(a,1),44(a,1,2)),rewrite([3(3),4305(6),112(10),114(10),429(15),2(12),3(12),428(12)])]*.

4553 x v ((x v (a ^ (y ^ z))) ^ (z v (a ^ (y ^ x)))) = x v (a ^ (y ^ z)). *[para(44(a,2),695(a,1,2,2)),rewrite([4300(9),870(13),23(12),216(12)])]*.

5121 (x v y) ^ ((z ^ y) v x) = (z ^ y) v x. *[para(159(a,1),698(a,1,1))]*.

5163 a ^ (c3 ^ (c1 v (x v c2))) = a ^ c3. *[para(23(a,1),712(a,1,2,2))]*.

5929 x ^ ((a ^ x) v (a ^ y)) = a ^ x. *[para(4(a,1),4488(a,1,2))]*.

6005 a ^ (x ^ (c3 ^ (c1 v (y v c2)))) = x ^ (a ^ c3). *[para(5163(a,1),21(a,1,2)),flip(a)]*.

6223 x v (a ^ (y ^ (x v z))) = x v (a ^ (y ^ z)). *[para(483(a,1),58(a,1,1)),rewrite([5(14),407(13),534(8),259(10),483(20),23(15),4(14),851(14),23(12),4300(11),4553(15)])]*.

6337 (a ^ x) v (a ^ ((x v (a ^ y)) ^ z)) = ((a ^ x) v (a ^ y)) ^ (x v (a ^ (z ^ ((a ^ x) v (a ^ y))))). *[para(5929(a,1),36(a,1,1)),rewrite([133(9),23(18),301(18),4(23)])]*.

6572 c1 v (c3 v (a ^ ((c1 v c2) ^ x))) = c1 v c3. *[para(83(a,1),90(a,1,2)),rewrite([5(10)])]*.

6603 (c1 v c3) ^ (x ^ a) = (c1 v c2) ^ (x ^ a). *[para(83(a,1),207(a,1)),rewrite([207(8)]),flip(a)]*.

7269 (x v y) ^ (x v (z ^ (u ^ (x v y)))) = x v (z ^ (u ^ (x v y))). *[para(217(a,1),160(a,1,2)),rewrite([2(6)])]*.

7441 (x ^ ((a ^ y) v (a ^ z))) v ((z v (a ^ y)) ^ x) = x ^ ((a ^ y) v z). *[para(112(a,1),68(a,1,2,1,2))]*.

8538 x ^ ((x ^ y) v (x ^ z)) = (x ^ y) v (x ^ z). *[para(6(a,1),261(a,1,1))]*.

8763 x v ((x v y) ^ (x v z)) = (x v y) ^ (x v z). *[para(7(a,1),276(a,1,1))]*.

9151 (x ^ (y ^ (z v u))) v ((w ^ (x ^ (y ^ z))) v v5) = (x ^ (y ^ (z v u))) v v5. *[para(280(a,1),216(a,1,2,1,2))]*.

9174 (x ^ (y ^ (z ^ u))) v (w v (y ^ (z ^ (u v v5)))) = w v (y ^ (z ^ (u v v5))). *[para(280(a,1),534(a,1,1,2))]*.

9221 (x ^ (y ^ z)) v (x ^ (u v y)) = x ^ (u v y). *[para(263(a,1),292(a,1,2,2)),rewrite([263(8)])]*.

9343 x ^ ((y ^ z) v (y ^ x)) = x ^ y. *[para(429(a,1),302(a,1,2)),rewrite([2(5),3(5),8538(4),429(8)])]*.

9373 x v (a ^ (y ^ (z v x))) = x v (a ^ (y ^ z)). *[para(302(a,1),58(a,1,1)),rewrite([5(15),297(14),9174(10),302(20),9151(16),4(11),23(12),4300(11),4553(15)])]*.

9434 x ^ ((y ^ z) v (z ^ x)) = x ^ z. *[para(2(a,1),9343(a,1,2,1))]*.

9435 x ^ ((y ^ z) v (x ^ y)) = x ^ y. *[para(2(a,1),9343(a,1,2,2))]*.

9874 x ^ ((y ^ z) v (x ^ (u v z))) = x ^ (u v z). *[para(321(a,1),9435(a,1,2,1))]*.

11144 x v (((x ^ y) v (x ^ z)) ^ u) = x. *[para(338(a,1),26(a,1)),rewrite([6(2)]),flip(a)]*.

11205 x v (y v (((x ^ z) v (y ^ u)) ^ w)) = x v y. *[para(338(a,1),262(a,1,2)),rewrite([152(3)]),flip(a)]*.

11222 x v (y ^ ((x ^ z) v (x ^ u))) = x. *[para(2(a,1),11144(a,1,2))]*.

11735 x ^ (y ^ ((x ^ z) v (x ^ u))) = y ^ ((x ^ z) v (x ^ u)). *[para(11222(a,1),30(a,1,2)),rewrite([2(5)])]*.

11820 (a ^ x) v (a ^ ((x v (a ^ y)) ^ z)) = ((a ^ x) v (a ^ y)) ^ (x v (z ^ ((a ^ x) v (a ^ y)))). *[back_rewrite(6337),rewrite([11735(22)])]*.

12718 x v ((y v x) ^ (y v z)) = x v y. *[para(483(a,1),353(a,1,2)),rewrite([23(5),4(4),8763(4),483(8)])]*.

12799 x v ((y v z) ^ (y v x)) = x v y. *[para(2(a,1),12718(a,1,2))]*.

12819 x v (((y ^ z) v x) ^ (u v z)) = x v (y ^ z). *[para(304(a,1),12718(a,1,2,2))]*.

12946 x v ((y v z) ^ ((u ^ z) v x)) = x v (u ^ z). *[para(304(a,1),12799(a,1,2,1))]*.

13031 (a ^ x) v (y ^ ((x v z) ^ u)) = ((a ^ x) v (y ^ u)) ^ (x v z). *[para(7(a,1),92(a,2,1,1,2)),rewrite([5(7),7(8),4(6),23(16),163(16)])]*.

13409 ((a ^ x) v (a ^ y)) ^ (x v (a ^ z)) = ((a ^ x) v (a ^ z)) ^ (x v (y ^ ((a ^ x) v (a ^ z)))). *[back_rewrite(11820),rewrite([13031(9)])].*

16801 x v (a ^ (x v y)) = x v (a ^ y). *[para(115(a,1),26(a,1,2)),rewrite([12946(8)]),flip(a)].*

16811 x v (a ^ (y v x)) = x v (a ^ y). *[para(115(a,1),133(a,1,2)),rewrite([12819(8)]),flip(a)].*

16931 (a ^ x) v (a ^ y) = a ^ (x v y). *[para(115(a,2),41(a,1,2,2)),rewrite([9874(10),9221(7),2(10),3(10),30(9),112(7),2(11),29(12),5121(12)]),flip(a)].*

17283 a ^ (x v (y ^ (a ^ (x v z)))) = a ^ ((x v y) ^ (x v (a ^ z))). *[back_rewrite(13409), rewrite([16931(5),3(7),16931(12),16931(15),3(16),7269(15)]),flip(a)].*

17360 (x ^ (a ^ (y v z))) v ((z v (a ^ y)) ^ x) = x ^ ((a ^ y) v z). *[back_rewrite(7441), rewrite([16931(5)])].*

17661 c1 v (a ^ c3) = c1 v (a ^ c2). *[para(17(a,1),16801(a,1,2)),rewrite([16801(7)]),flip(a)].*

17662 a ^ (x v (a ^ y)) = a ^ (x v y). *[para(16801(a,1),30(a,1,2)),rewrite([3(7),380(6)])].*

17667 x v (a ^ ((y ^ x) v z)) = x v (a ^ z). *[para(16801(a,1),133(a,1,2)),rewrite([133(5)]),flip(a)].*

17670 a ^ (x ^ (y v (a ^ z))) = x ^ (a ^ (y v z)). *[para(16801(a,1),149(a,1,2,2)),rewrite([3(8),582(7)])].*

17727 (x ^ a) v (a ^ y) = a ^ ((x ^ a) v y). *[para(16801(a,1),9434(a,1,2)),rewrite([380(9),2(10)])].*

17951 a ^ (x v (y ^ (a ^ (x v z)))) = a ^ ((x v y) ^ (x v z)). *[back_rewrite(17283),rewrite([17670(14),21(12)])].*

18506 a ^ (c1 v (c2 v c3)) = a ^ (c1 v c2). *[para(17661(a,1),16811(a,1,2,2)),rewrite([17662(10),16931(9),23(6),4(5),4(14),16931(14),17(12)])].*

18675 a ^ ((x ^ a) v y) = a ^ (x v y). *[para(2(a,1),16931(a,1,1)),rewrite([17727(5)])].*

18680 (a ^ x) v (y ^ (a ^ z)) = a ^ (x v (y ^ z)). *[para(21(a,1),16931(a,1,2))].*

18682 a ^ ((a ^ x) v y) = a ^ (x v y). *[para(112(a,1),16931(a,1,1)),rewrite([16931(5)]),flip(a)].*

18698 a ^ (x v (y ^ (a ^ z))) = a ^ (x v (y ^ z)). *[para(170(a,1),16931(a,1,2)),rewrite([18680(6)]),flip(a)].*

18803 (x ^ a) v (a ^ y) = a ^ (x v y). *[back_rewrite(17727),rewrite([18675(10)])].*

19197 a ^ (x v (y ^ (x v z))) = a ^ ((x v y) ^ (x v z)). *[back_rewrite(17951),rewrite([18698(7)])].*

19661 a ^ ((c1 v (c2 v c3)) ^ x) = a ^ ((c1 v c2) ^ x). *[para(18506(a,1),3(a,1,1)),rewrite([3(6)]),flip(a)].*

19740 a ^ (x ^ ((a ^ y) v z)) = a ^ (x ^ (y v z)). *[para(68(a,1),18675(a,2,2)),rewrite([2(7),17670(7),17360(10)])].*

19911 a ^ (((a ^ x) v y) ^ z) = a ^ ((x v y) ^ z). *[para(18682(a,1),3(a,1,1)),rewrite([3(4)]),flip(a)].*

21622 (a ^ x) v (x ^ y) = x ^ ((a ^ x) v y). *[para(6(a,1),174(a,1,2,1)),rewrite([6(9),2(8)])].*

23068 (x ^ (y ^ z)) v (x ^ (z v u)) = (z v u) ^ x. *[para(514(a,1),429(a,1,1))].*

25533 a ^ (x ^ ((x ^ y) v z)) = a ^ (x ^ (y v z)). *[para(6(a,1),236(a,1,2,2,2)),rewrite([1688(7),19740(7)])].*

30690 c1 v (a ^ (c3 v ((c1 v c2) ^ x))) = c1 v (a ^ c2). *[para(6572(a,1),16801(a,1,2,2)), rewrite([17(6),16801(7),17662(16)]),flip(a)].*

30792 a ^ (x v ((c1 v c2) ^ y)) = a ^ ((x v (c1 v c2)) ^ (x v y)). *[para(6603(a,1),267(a,2,1,2,2)),rewrite([2(9),16801(10),17670(11),21(9),83(9),18803(10),19197(8),2(15),21(16),112(17),6223(16),83(21),2(24),23068(25),2(18),2(19),3(19),904(18),17662(17)]),flip(a)].*

30797 c1 v (a ^ ((c1 v c2) ^ (c3 v x))) = c1 v (a ^ c2). *[back_rewrite(30690),rewrite([30792(9),23(7),4(6),19661(11)])].*

50699 x v (a ^ ((x v y) ^ z)) = x v (a ^ (y ^ z)). *[para(512(a,1),17667(a,1,2,2)),rewrite([19911(7),9373(6),9373(10)])].*

50705 c1 v (a ^ (c2 ^ (c3 v x))) = c1 v (a ^ c2). *[back_rewrite(30797),rewrite([50699(10)])].*

50709 c1 v (a ^ (c2 ^ c3)) = c1 v (a ^ c2). *[para(6(a,1),50705(a,1,2,2,2))].*

50727 c1 v (c2 ^ (c3 v (a ^ c2))) = c1 v (c2 ^ c3). *[para(50709(a,1),159(a,1,2)),rewrite([23(9),4(8),21622(8),4(7)])]*.
50801 a ^ (x v (y ^ z)) = a ^ ((x v z) ^ (x v y)). *[para(516(a,1),18682(a,1,2)),rewrite([4(9),2786(9),16931(8),21(7),21(8),112(7),112(7),9373(11),17662(11)]),flip(a)]*.
50871 a ^ (c2 ^ (x v (c1 v (c2 ^ c3)))) = a ^ c2. *[para(50727(a,1),461(a,1,2,2,2)),rewrite([16931(8),3(13),21(14),28(13),16931(18),21(17),7(16)])]*.
50874 a ^ (c2 ^ (c1 v ((c2 ^ c3) v x))) = a ^ c2. *[para(4(a,1),50871(a,1,2,2)),rewrite([5(8)])]*.
50901 a ^ (c2 ^ (c3 v (x v c1))) = a ^ c2. *[para(185(a,1),50874(a,1,2,2)),rewrite([25533(10)])]*.
53798 a ^ (c3 ^ ((c1 ^ c3) v (x v c2))) = a ^ c3. *[para(6005(a,1),566(a,1,2,2)),rewrite([2(4),4(16),112(21),2(20),3(20),17670(20),50801(17),2(18),83(19),160(18),21(15),352(15),2(12),3(12),19740(12),5163(19),112(15)])]*.
53820 a ^ (c3 ^ (c2 v (c1 ^ c3))) = a ^ c3. *[para(32(a,1),53798(a,1,2,2,2)),rewrite([4(7)])]*.
54317 c3 v (a ^ (c2 ^ (x v c1))) = c3 v (a ^ c2). *[para(50901(a,1),6223(a,1,2)),flip(a)]*.
54322 c2 v (a ^ (c1 ^ c3)) = c2 v (a ^ c3). *[para(53820(a,1),6223(a,1,2)),rewrite([114(12)]),flip(a)]*.
54473 c3 v (a ^ (c1 ^ c2)) = c3 v (a ^ c2). *[para(32(a,1),54317(a,1,2,2,2)),rewrite([2(5)])]*.
60183 a ^ (x ^ (y v (z ^ u))) = x ^ (a ^ ((y v u) ^ (y v z))). *[para(11205(a,1),458(a,1,2,2,2)),rewrite([280(8),2(4),50801(4)]),flip(a)]*.
73833 (c1 ^ c3) v (c1 ^ (c2 v (a ^ c3))) != c1 ^ (c2 v c3). *[para(1373(a,1),19(a,1)),rewrite([112(9),4(10),54322(10),2(14),21(14),4(17),90(17),2(10)])]*.
73856 $F. *[para(1373(a,1),73833(a,1)),rewrite([112(17),60183(16),4(15),2(16),28(17),21(14),4(16),9373(16),54473(14),2(22),21(22),847(23),4(19),90(19),2(14),15(15)]),xx(a)]*.

============================== **end of proof** ==========================

============================== **STATISTICS** ============================

```
Given=1611. Generated=4107893. Kept=73850. proofs=1.
Usable=1438. Sos=19999. Demods=19736. Limbo=0, Disabled=52427. Hints=0.
Kept_by_rule=0, Deleted_by_rule=5024.
Forward_subsumed=2472886. Back_subsumed=870.
Sos_limit_deleted=1556132. Sos_displaced=38067. Sos_removed=0.
New_demodulators=64146 (6 lex), Back_demodulated=13474. Back_unit_deleted=0.
Demod_attempts=108087980. Demod_rewrites=13862951.
Res_instance_prunes=0. Para_instance_prunes=0. Basic_paramod_prunes=0.
Nonunit_fsub_feature_tests=0. Nonunit_bsub_feature_tests=0.
Megabytes=107.30.
User_CPU=222.55, System_CPU=1.51, Wall_clock=391.
```

============================== **end of statistics** =====================

============================== **end of search** =========================

### THEOREM PROVED

Just for comparison, see how much simpler the proof gets when RL1 is assumed:

**formulas(assumptions)**.
```
x ^ y = y ^ x.
(x ^ y) ^ z = x ^ (y ^ z).
x v y = y v x.
(x v y) v z = x v (y v z).
x v (x ^ y) = x.
```

```
x ^ (x v y) = x.
UpMe(x,y,z) = x ^ (y v z).
LoMe(x,y,z) = (x ^ y) v (x ^ z).
UpJo(x,y,z) = (x v y) ^ (x v z).
LoJo(x,y,z) = x v (y ^ z).
UpMe(a ^ x1,y1,z1) v (y1 ^ z1) = (((a ^ x1) ^ y1) v z1) ^ (((a ^ x1) ^ z1) v y1).
LoMe(x,y,z) = UpMe(x,UpMe(y,x,z),UpMe(z,x,y)).
UpMe(x,y,z) = UpMe(x,y,a ^ z) v UpMe(x,z,a ^ y).
```
**end_of_list**.

**formulas**(**goals**).

```
(all x2 all y2 all z2 (UpMe(a,x2,y2) = UpMe(a,x2,z2) -> UpMe(x2,y2,z2) = LoMe(x2,y2,z2)))
    .
```
**end_of_list**.

.....

============================== **PROOF** =================================

% **Proof** 1 at 88.76 (+ 0.59) seconds.
% **Length of proof** is 38.
% **Level of proof** is 8.
% Maximum clause weight is 29.
% Given clauses 936.

1 (all x2 all y2 all z2 (UpMe(a,x2,y2) = UpMe(a,x2,z2) -> UpMe(x2,y2,z2) = LoMe(x2,y2,z2) )) # *label*(*non_clause*) # *label*(*goal*). *[goal]*.
2 x ^ y = y ^ x. *[assumption]*.
3 (x ^ y) ^ z = x ^ (y ^ z). *[assumption]*.
4 x v y = y v x. *[assumption]*.
5 (x v y) v z = x v (y v z). *[assumption]*.
6 x v (x ^ y) = x. *[assumption]*.
7 x ^ (x v y) = x. *[assumption]*.
8 UpMe(x,y,z) = x ^ (y v z). *[assumption]*.
9 LoMe(x,y,z) = (x ^ y) v (x ^ z). *[assumption]*.
12 UpMe(a ^ x,y,z) v (y ^ z) = (((a ^ x) ^ y) v z) ^ (((a ^ x) ^ z) v y). *[assumption]*.
13 (x ^ y) v (a ^ (z ^ (x v y))) = ((a ^ (z ^ x)) v y) ^ ((a ^ (z ^ y)) v x). *[copy(12), rewrite([8(3),3(4),4(6),3(9),3(13)])]*.
14 LoMe(x,y,z) = UpMe(x,UpMe(y,x,z),UpMe(z,x,y)). *[assumption]*.
15 (x ^ y) v (x ^ z) = x ^ ((y ^ (x v z)) v (z ^ (x v y))). *[copy(14),rewrite([9(1),8(4),8(6) ,8(8)])]*.
16 UpMe(x,y,z) = UpMe(x,y,a ^ z) v UpMe(x,z,a ^ y). *[assumption]*.
17 (x ^ (y v (a ^ z))) v (x ^ (z v (a ^ y))) = x ^ (y v z). *[copy(16),rewrite([8(1),8(5),8(9)]), flip(a)]*.
18 UpMe(a,c1,c3) = UpMe(a,c1,c2). *[deny(1)]*.
19 a ^ (c1 v c3) = a ^ (c1 v c2). *[copy(18),rewrite([8(4),8(9)])]*.
20 LoMe(c1,c2,c3) != UpMe(c1,c2,c3). *[deny(1)]*.
21 (c1 ^ c2) v (c1 ^ c3) != c1 ^ (c2 v c3). *[copy(20),rewrite([9(4),8(11)])]*.
23 x ^ (y ^ z) = y ^ (x ^ z). *[para(2(a,1),3(a,1,1)),rewrite([3(2)])]*.
28 x v ((x ^ y) v z) = x v z. *[para(6(a,1),5(a,1,1)),flip(a)]*.
30 x ^ ((x v y) ^ z) = x ^ z. *[para(7(a,1),3(a,1,1)),flip(a)]*.
32 x ^ (y v x) = x. *[para(4(a,1),7(a,1,2))]*.

```
34 x v x = x. [para(7(a,1),6(a,1,2))].
67 x ^ ((y ^ (x v z)) v (z ^ (x v y))) = (y ^ x) v (x ^ z). [para(2(a,1),15(a,1,1)),flip(a)].
132 a ^ (x ^ (c1 v c3)) = x ^ (a ^ (c1 v c2)). [para(19(a,1),23(a,1,2)),flip(a)].
143 x v (y v x) = y v x. [para(34(a,1),5(a,2,2)),rewrite([4(2)])].
183 x ^ (y v (z v x)) = x. [para(5(a,1),32(a,1,2))].
186 ((a ^ x) v y) ^ ((a ^ (x ^ y)) v (z v x)) = (a ^ x) v (y ^ (z v x)). [para(32(a,1),13(a
    ,2,2,1,2)),rewrite([183(6),4(5),2(14)]),flip(a)].
189 x ^ (y ^ (z v x)) = y ^ x. [para(32(a,1),23(a,1,2)),flip(a)].
231 x v (y v (z v x)) = y v (z v x). [para(5(a,1),143(a,1,2)),rewrite([5(5)])].
312 (x ^ y) v (z v x) = z v x. [para(143(a,1),28(a,2)),rewrite([231(4)])].
401 a ^ (c3 ^ (c1 v c2)) = a ^ c3. [para(19(a,1),189(a,1,2)),rewrite([23(7)])].
553 (x ^ (y ^ z)) v (u v y) = u v y. [para(23(a,1),312(a,1,1))].
562 (a ^ x) v (y ^ (z v x)) = ((a ^ x) v y) ^ (z v x). [back_rewrite(186),rewrite([553(8)]),flip(a
    )].
721 (x ^ (y v (a ^ c3))) v (x ^ ((c3 ^ (c1 v c2)) v (a ^ y))) = x ^ (y v (c3 ^ (c1 v c2))
    ). [para(401(a,1),17(a,1,1,2,2))].
51633 (c1 ^ c2) v (c1 ^ c3) = c1 ^ (c2 v c3). [para(67(a,1),721(a,2)),rewrite([4(10),562(10),4(6)
    ,2(10),30(11),132(20),23(20),32(19),4(17),562(17),4(13),2(17),30(18),17(15),2(8)]),flip(a)].
51634 $F. [resolve(51633,a,21,a)].
```

============================== **end of proof** ==========================

============================== **STATISTICS** ============================

```
Given=936. Generated=1315923. Kept=51627. proofs=1.
Usable=861. Sos=19999. Demods=18987. Limbo=1, Disabled=30780. Hints=0.
Kept_by_rule=0, Deleted_by_rule=3409.
Forward_subsumed=884193. Back_subsumed=793.
Sos_limit_deleted=376694. Sos_displaced=18619. Sos_removed=0.
New_demodulators=43355 (6 lex), Back_demodulated=11351. Back_unit_deleted=0.
Demod_attempts=32232244. Demod_rewrites=4313279.
Res_instance_prunes=0. Para_instance_prunes=0. Basic_paramod_prunes=0.
Nonunit_fsub_feature_tests=0. Nonunit_bsub_feature_tests=0.
Megabytes=87.32.
User_CPU=88.76, System_CPU=0.59, Wall_clock=143.
```

============================== **end of statistics** =====================

============================== **end of search** =========================

**THEOREM PROVED**

**Appendix C. Theorem 3.4, Clause 9:**

**formulas**(**assumptions**).
```
x ^ y = y ^ x.
(x ^ y) ^ z = x ^ (y ^ z).
x v y = y v x.
(x v y) v z = x v (y v z).
x v (x ^ y) = x.
x ^ (x v y) = x.
UpMe(x,y,z) = x ^ (y v z).
LoMe(x,y,z) = (x ^ y) v (x ^ z).
UpJo(x,y,z) = (x v y) ^ (x v z).
LoJo(x,y,z) = x v (y ^ z).
UpMe(a ^ x1,y1,z1) v (y1 ^ z1) = (((a ^ x1) ^ y1) v z1) ^ (((a ^ x1) ^ z1) v y1).
UpMe(x,y,z) = UpMe(x,y,a ^ z) v UpMe(x,z,a ^ y).
```
**end_of_list**.

**formulas**(**goals**).
```
(all x2 all y2 all z2 (UpMe(a,x2,y2) = UpMe(a,x2,z2) -> (UpMe(a,x2,y2) = UpMe(a,y2,z2) ->

UpJo(x2,y2,z2) = LoJo(x2,y2,z2)))).
```
**end_of_list**.

......

============================== **PROOF** =================================

% **Proof** 1 at 963.33 (+ 15.06) seconds.
% **Length of proof** is 196.
% **Level of proof** is 20.
% Maximum clause weight is 47.000.
% Given clauses 4826.

1 (all x2 all y2 all z2 (UpMe(a,x2,y2) = UpMe(a,x2,z2) -> (UpMe(a,x2,y2) = UpMe(a,y2,z2) -> UpJo(x2,y2,z2) = LoJo(x2,y2,z2)))) # *label*(*non_clause*) # *label*(*goal*). *[goal]*.
2 x ^ y = y ^ x. *[assumption]*.
3 (x ^ y) ^ z = x ^ (y ^ z). *[assumption]*.
4 x v y = y v x. *[assumption]*.
5 (x v y) v z = x v (y v z). *[assumption]*.
6 x v (x ^ y) = x. *[assumption]*.
7 x ^ (x v y) = x. *[assumption]*.
8 UpMe(x,y,z) = x ^ (y v z). *[assumption]*.
9 LoMe(x,y,z) = (x ^ y) v (x ^ z). *[assumption]*.
10 UpJo(x,y,z) = (x v y) ^ (x v z). *[assumption]*.
11 LoJo(x,y,z) = x v (y ^ z). *[assumption]*.
12 UpMe(a ^ x,y,z) v (y ^ z) = (((a ^ x) ^ y) v z) ^ (((a ^ x) ^ z) v y). *[assumption]*.
13 (x ^ y) v (a ^ (z ^ (x v y))) = ((a ^ (z ^ x)) v y) ^ ((a ^ (z ^ y)) v x). *[copy(12), rewrite([8(3),3(4),4(6),3(9),3(13)])]*.
14 UpMe(x,y,z) = UpMe(x,y,a ^ z) v UpMe(x,z,a ^ y). *[assumption]*.
15 (x ^ (y v (a ^ z))) v (x ^ (z v (a ^ y))) = x ^ (y v z). *[copy(14),rewrite([8(1),8(5),8(9)]), flip(a)]*.

16 UpMe(a,c1,c3) = UpMe(a,c1,c2). *[deny(1)]*.
17 a ^ (c1 v c3) = a ^ (c1 v c2). *[copy(16),rewrite([8(4),8(9)])]*.
18 UpMe(a,c2,c3) = UpMe(a,c1,c2). *[deny(1)]*.
19 a ^ (c2 v c3) = a ^ (c1 v c2). *[copy(18),rewrite([8(4),8(9)])]*.
20 LoJo(c1,c2,c3) != UpJo(c1,c2,c3). *[deny(1)]*.
21 c1 v (c2 ^ c3) != (c1 v c2) ^ (c1 v c3). *[copy(20),rewrite([11(4),10(9)])]*.
23 x ^ (y ^ z) = y ^ (x ^ z). *[para(2(a,1),3(a,1,1)),rewrite([3(2)])]*.
25 x v (y v z) = y v (x v z). *[para(4(a,1),5(a,1,1)),rewrite([5(2)])]*.
26 x v (y ^ x) = x. *[para(2(a,1),6(a,1,2))]*.
27 (x ^ y) v (x ^ (y ^ z)) = x ^ y. *[para(3(a,1),6(a,1,2))]*.
28 x v ((x ^ y) v z) = x v z. *[para(6(a,1),5(a,1,1)),flip(a)]*.
29 x v (y v ((x v y) ^ z)) = x v y. *[para(6(a,1),5(a,1)),flip(a)]*.
30 x ^ ((x v y) ^ z) = x ^ z. *[para(7(a,1),3(a,1,1)),flip(a)]*.
31 x ^ (y ^ ((x ^ y) v z)) = x ^ y. *[para(7(a,1),3(a,1)),flip(a)]*.
32 x ^ (y v x) = x. *[para(4(a,1),7(a,1,2))]*.
33 (x v y) ^ (x v (y v z)) = x v y. *[para(5(a,1),7(a,1,2))]*.
34 x v x = x. *[para(7(a,1),6(a,1,2))]*.
35 x ^ x = x. *[para(6(a,1),7(a,1,2))]*.
38 (x ^ y) v (a ^ ((x v y) ^ z)) = ((a ^ (z ^ x)) v y) ^ ((a ^ (z ^ y)) v x). *[para(2(a,1),13(a,1,2,2))]*.
39 (x ^ y) v (a ^ (z ^ (x v y))) = ((a ^ (x ^ z)) v y) ^ ((a ^ (z ^ y)) v x). *[para(2(a,1),13(a,2,1,1,2))]*.
43 (x ^ (y ^ z)) v (a ^ (u ^ ((x ^ y) v z))) = ((a ^ (u ^ (x ^ y))) v z) ^ ((a ^ (u ^ z)) v (x ^ y)). *[para(3(a,1),13(a,1,1))]*.
46 (x ^ y) v (a ^ (z ^ (x v y))) = (y v (a ^ (z ^ x))) ^ ((a ^ (z ^ y)) v x). *[para(4(a,1),13(a,2,1))]*.
57 ((a ^ x) v y) ^ (x v ((a ^ (x ^ y)) v z)) = (a ^ x) v ((x v z) ^ y). *[para(7(a,1),13(a,2,1,1,2)),rewrite([5(5),7(6),4(5),25(13)]),flip(a)]*.
60 (((a ^ (x ^ y)) v z) ^ u) v (((a ^ (x ^ y)) v z) ^ ((a ^ (x ^ z)) v y)) = (y ^ z) v ((a ^ (x ^ (y v z))) v (((a ^ (x ^ y)) v z) ^ u)). *[para(13(a,2),9(a,2,2)),rewrite([9(9),25(27),4(26)])]*.
67 (x ^ (y v (z ^ a))) v (x ^ (z v (a ^ y))) = x ^ (y v z). *[para(2(a,1),15(a,1,1,2,2))]*.
86 (a ^ (c1 v c2)) v (a ^ (x ^ (a v (c1 v c3)))) = a ^ (c1 v (c3 v (a ^ (x ^ a)))). *[para(17(a,1),13(a,1,1)),rewrite([25(22),4(21),4(30),6(30),2(24)])]*.
91 (x ^ (c1 v (c3 v (a ^ y)))) v (x ^ (y v (a ^ (c1 v c2)))) = x ^ (c1 v (c3 v y)). *[para(17(a,1),15(a,1,2,2,2)),rewrite([5(6),5(19)])]*.
93 (a ^ (c1 v c2)) v (a ^ (x ^ (a v (c2 v c3)))) = a ^ (c2 v (c3 v (a ^ (x ^ a)))). *[para(19(a,1),13(a,1,1)),rewrite([25(22),4(21),4(30),6(30),2(24)])]*.
98 (x ^ (c2 v (c3 v (a ^ y)))) v (x ^ (y v (a ^ (c1 v c2)))) = x ^ (c2 v (c3 v y)). *[para(19(a,1),15(a,1,2,2,2)),rewrite([5(6),5(19)])]*.
99 x v (y ^ (x ^ z)) = x. *[para(23(a,1),6(a,1,2))]*.
100 x ^ (y ^ (x v z)) = y ^ x. *[para(7(a,1),23(a,1,2)),flip(a)]*.
110 a ^ (x ^ (c1 v c3)) = x ^ (a ^ (c1 v c2)). *[para(17(a,1),23(a,1,2)),flip(a)]*.
111 a ^ (x ^ (c2 v c3)) = x ^ (a ^ (c1 v c2)). *[para(19(a,1),23(a,1,2)),flip(a)]*.
113 (a ^ (c1 v c2)) v (x ^ a) = a ^ (c2 v (c3 v (a ^ (x ^ a)))). *[back_rewrite(93),rewrite([100(13)])]*.
114 a ^ (c2 v (c3 v (a ^ (x ^ a)))) = a ^ (c1 v (c3 v (a ^ (x ^ a)))). *[back_rewrite(86),rewrite([100(13),113(8)])]*.
117 (a ^ (c1 v c2)) v (x ^ a) = a ^ (c1 v (c3 v (a ^ (x ^ a)))). *[back_rewrite(113),rewrite([114(18)])]*.
118 x v (x v y) = x v y. *[para(34(a,1),5(a,1,1)),flip(a)]*.
120 x v (y v x) = y v x. *[para(34(a,1),5(a,2,2)),rewrite([4(2)])]*.

```
124 x ^ (x ^ y) = x ^ y. [para(35(a,1),3(a,1,1)),flip(a)].
126 x ^ (y ^ x) = y ^ x. [para(35(a,1),3(a,2,2)),rewrite([2(2)])].
127 (x ^ y) v (a ^ (x v y)) = ((a ^ x) v y) ^ ((a ^ y) v x). [para(35(a,1),13(a,1,2,2)),rewrite([2(8),7(8),2(11),32(11)])].
128 (x ^ y) v (a ^ y) = y ^ ((a ^ y) v x). [para(35(a,1),13(a,2,2,1,2)),rewrite([32(4),4(8),99(8)])].
144 (a ^ (c1 v c2)) v (x ^ a) = a ^ (c1 v (c3 v (x ^ a))). [back_rewrite(117),rewrite([126(15)])].
148 x v (y ^ (z ^ x)) = x. [para(3(a,1),26(a,1,2))].
149 x v ((y ^ x) v z) = x v z. [para(26(a,1),5(a,1,1)),flip(a)].
150 x v (y v (z ^ (x v y))) = x v y. [para(26(a,1),5(a,1)),flip(a)].
157 (x ^ y) v (x ^ (z ^ y)) = x ^ y. [para(23(a,1),26(a,1,2))].
158 x ^ ((y v x) ^ z) = x ^ z. [para(32(a,1),3(a,1,1)),flip(a)].
159 x ^ (y ^ (z v (x ^ y))) = x ^ y. [para(32(a,1),3(a,1)),flip(a)].
160 x ^ (y v (z v x)) = x. [para(5(a,1),32(a,1,2))].
166 x ^ (y ^ (z v x)) = y ^ x. [para(32(a,1),23(a,1,2)),flip(a)].
169 x v (y v (x ^ z)) = y v x. [para(6(a,1),25(a,1,2)),flip(a)].
170 x ^ (y v (x v z)) = x. [para(25(a,1),7(a,1,2))].
176 x v (y v (z ^ x)) = y v x. [para(26(a,1),25(a,1,2)),flip(a)].
177 (x v y) ^ (x v (z v y)) = x v y. [para(25(a,1),32(a,1,2))].
180 x v ((y ^ (x ^ z)) v u) = x v u. [para(99(a,1),5(a,1,1)),flip(a)].
191 (a ^ x) v ((x v y) ^ z) = ((a ^ x) v z) ^ (x v y). [back_rewrite(57),rewrite([180(8)]),flip(a)].
202 x v (y v (z v x)) = y v (z v x). [para(5(a,1),120(a,1,2)),rewrite([5(5)])].
213 (x ^ y) v ((x ^ (y ^ z)) v u) = (x ^ y) v u. [para(27(a,1),5(a,1,1)),flip(a)].
224 x ^ (y ^ (z ^ x)) = y ^ (z ^ x). [para(3(a,1),126(a,1,2)),rewrite([3(5)])].
230 (x ^ y) v (y ^ x) = x ^ y. [para(126(a,1),27(a,1,2))].
232 x v (y ^ (z ^ (u ^ x))) = x. [para(3(a,1),148(a,1,2,2))].
233 x v ((y ^ (z ^ x)) v u) = x v u. [para(148(a,1),5(a,1,1)),flip(a)].
239 c1 v (c3 v (x ^ (a ^ (c1 v c2)))) = c1 v c3. [para(17(a,1),148(a,1,2,2)),rewrite([5(10)])].
249 x ^ (y ^ (z v (u v (x ^ y)))) = x ^ y. [para(160(a,1),3(a,1)),flip(a)].
250 x ^ (y v (z v (u v x))) = x. [para(5(a,1),160(a,1,2,2))].
267 (a ^ x) v (y ^ (x v z)) = (x v z) ^ ((a ^ x) v y). [para(170(a,1),13(a,1,2,2)),rewrite([4(5),25(10),180(10),7(9)])].
277 x v (((a ^ (y ^ x)) v z) ^ ((a ^ (y ^ z)) v x)) = x v (a ^ (y ^ (x v z))). [para(13(a,1),28(a,1,2))].
279 (x v y) ^ ((x ^ z) v y) = (x ^ z) v y. [para(28(a,1),32(a,1,2)),rewrite([2(4)])].
280 x v (y v ((x ^ z) v u)) = y v (x v u). [para(28(a,1),25(a,1,2)),flip(a)].
281 (x ^ y) v (z v x) = z v x. [para(120(a,1),28(a,2)),rewrite([202(4)])].
293 x ^ (y ^ ((x v z) ^ u)) = y ^ (x ^ u). [para(30(a,1),23(a,1,2)),flip(a)].
294 (x ^ y) v ((x v z) ^ y) = (x v z) ^ y. [para(30(a,1),26(a,1,2)),rewrite([4(4)])].
295 (x v y) ^ (z ^ x) = z ^ x. [para(126(a,1),30(a,2)),rewrite([224(4)])].
296 ((x v y) ^ z) v (u ^ (x ^ z)) = (x v y) ^ z. [para(30(a,1),148(a,1,2,2))].
298 x ^ (y ^ (z ^ (x v u))) = y ^ (z ^ x). [para(3(a,1),100(a,1,2)),rewrite([3(6)])].
310 (x ^ y) v (x ^ (y v z)) = x ^ (y v z). [para(100(a,1),26(a,1,2)),rewrite([4(4)])].
315 x v (((a ^ (y ^ z)) v x) ^ ((a ^ (y ^ x)) v z)) = x v (a ^ (y ^ (z v x))). [para(13(a,1),149(a,1,2))].
321 (x v y) ^ ((z ^ x) v y) = (z ^ x) v y. [para(149(a,1),32(a,1,2)),rewrite([2(4)])].
322 x v (y v ((z ^ x) v u)) = y v (x v u). [para(149(a,1),25(a,1,2)),flip(a)].
323 (x ^ y) v (z v y) = z v y. [para(120(a,1),149(a,2)),rewrite([202(4)])].
338 (x ^ y) v ((z v x) ^ y) = (z v x) ^ y. [para(158(a,1),26(a,1,2)),rewrite([4(4)])].
340 (x v y) ^ (z ^ y) = z ^ y. [para(126(a,1),158(a,2)),rewrite([224(4)])].
```

357 x v (y v (((x ^ z) v y) ^ u)) = x v y. [para(29(a,1),28(a,1,2)),rewrite([28(3)]),flip(a)].
372 (x ^ y) v (x ^ (z v y)) = x ^ (z v y). [para(166(a,1),26(a,1,2)),rewrite([4(4)])].
390 x v (y v (z v (u ^ (x v y)))) = z v (x v y). [para(176(a,1),5(a,1)),flip(a)].
391 x v (y v (z v (u ^ x))) = y v (z v x). [para(5(a,1),176(a,1,2)),rewrite([5(6)])].
400 (x v y) ^ (x v (z ^ y)) = x v (z ^ y). [para(176(a,1),32(a,1,2)),rewrite([2(4)])].
427 (a ^ x) v (((a ^ x) v y) ^ z) = ((a ^ x) v z) ^ ((a ^ x) v y). [para(31(a,1),13(a,2,1,1)), rewrite([5(9),31(11),4(7),25(17),213(17)])].
444 (x ^ y) v (z v (y ^ x)) = z v (y ^ x). [para(230(a,1),5(a,2,2)),rewrite([4(4)])].
449 (x ^ (y ^ z)) v (y ^ x) = x ^ y. [para(230(a,1),28(a,2)),rewrite([3(3),444(6)])].
451 x v ((y ^ (z ^ (u ^ x))) v w) = x v w. [para(232(a,1),5(a,1,1)),flip(a)].
479 a ^ (x ^ ((y v z) ^ (u v (w v (((a ^ (x ^ y)) v z) ^ ((a ^ (x ^ z)) v y)))))) = a ^ ( x ^ (y v z)). [para(13(a,1),250(a,1,2,2,2)),rewrite([3(16),3(15)])].
504 (x v y) ^ (y v (x v z)) = y v x. [para(4(a,1),33(a,1,1))].
514 (x v y) ^ (y v x) = x v y. [para(120(a,1),33(a,1,2))].
516 (x ^ (y ^ z)) v (u v (x ^ y)) = u v (x ^ y). [para(3(a,1),281(a,1,1))].
537 (x ^ y) v (a ^ (z ^ (u v y))) = (y v u) ^ ((a ^ (z ^ (y v u))) v (x ^ y)). [para(295(a ,1),13(a,1,1)),rewrite([5(5),176(5),25(18),451(18),2(14)])].
556 (x ^ (y ^ z)) v (u v z) = u v z. [para(3(a,1),323(a,1,1))].
557 (x ^ y) v (z v (u v y)) = z v (u v y). [para(5(a,1),323(a,1,2)),rewrite([5(6)])].
606 (x v y) ^ (z ^ (x v (u ^ y))) = z ^ (x v (u ^ y)). [para(176(a,1),340(a,1,1))].
719 x v (y v (z ^ (y v x))) = x v y. [para(514(a,1),148(a,1,2,2)),rewrite([5(4)])].
722 (x v (y v z)) ^ (z v y) = z v y. [para(514(a,1),340(a,1,2)),rewrite([514(7)])].
745 (x v y) ^ (y v (z ^ (x v y))) = y v (z ^ (x v y)). [para(150(a,1),32(a,1,2)),rewrite([2(5)])] .
890 (x ^ (y ^ z)) v (x ^ (y ^ (z v u))) = x ^ (y ^ (z v u)). [para(100(a,1),157(a,1,2,2)), rewrite([4(6)])].
898 (x ^ (y v z)) v (x ^ (y v (z v u))) = x ^ (y v (z v u)). [para(33(a,1),157(a,1,2,2)), rewrite([4(6)])].
909 (a ^ x) v (y ^ (z v (a ^ x))) = (z v (a ^ x)) ^ ((a ^ x) v y). [para(159(a,1),13(a,2,2,1) ),rewrite([249(11),4(7),516(14)])].
921 (x ^ y) v (z v (y ^ (u v (x ^ y)))) = z v (y ^ (u v (x ^ y))). [para(159(a,1),323(a,1,1)) ].
941 (a ^ ((x v y) ^ z)) v ((a ^ (z ^ x)) v y) = (a ^ (z ^ x)) v y. [para(38(a,2),6(a,1,2)), rewrite([25(11),4(10),557(11)])].
964 (a ^ (c1 v c2)) v (a ^ x) = a ^ (c1 v (c3 v (x ^ a))). [para(17(a,1),38(a,1,1)),rewrite([ 30(13),126(12),25(14),4(13),110(20),4(22),99(22),2(16)])].
1108 (x v y) ^ (x v ((x v y) ^ z)) = x v ((x v y) ^ z). [para(29(a,1),177(a,1,2)),rewrite([2(5)]) ].
1243 a ^ (c2 v (c3 v (a ^ x))) = a ^ (c1 v (c3 v (x ^ a))). [para(19(a,1),39(a,1,1)),rewrite([ 100(13),144(8),124(12),25(14),4(13),111(20),4(22),99(22),2(16)]),flip(a)].
2810 (a ^ x) v (y ^ (z ^ (x v u))) = (x v u) ^ ((a ^ x) v (y ^ z)). [para(7(a,1),43(a,2,2,1,2) ),rewrite([170(8),4(6),25(12),180(12)])].
4331 (x ^ y) v (a ^ (z ^ (x v y))) = (y v (a ^ (z ^ x))) ^ (x v (a ^ (z ^ y))). [para(4(a ,1),46(a,2,2))].
4337 (a ^ x) v (y ^ a) = a ^ (x v (y ^ a)). [para(46(a,1),9(a,2)),rewrite([9(5),100(7),126(9) ,4(13),6(13),2(10)])].
4352 a ^ (c1 v (c3 v (x ^ a))) = a ^ (c1 v (c2 v (x ^ a))). [para(17(a,1),46(a,1,1)),rewrite([ 100(13),4337(8),5(7),126(15),5(14),110(20),4(22),99(22),2(16)]),flip(a)].
4586 x v ((x v (a ^ (y ^ z))) ^ (z v (a ^ (y ^ x)))) = x v (a ^ (y ^ z)). [para(46(a,2),719( a,1,2,2)),rewrite([4331(9),909(13),25(12),233(12)])].
5088 a ^ (c2 v (c3 v (a ^ x))) = a ^ (c1 v (c2 v (x ^ a))). [back_rewrite(1243),rewrite([ 4352(16)])].

5089 (a ^ (c1 v c2)) v (a ^ x) = a ^ (c1 v (c2 v (x ^ a))). *[back_rewrite(964),rewrite([4352(16)])]*.
5201 (x v y) ^ ((z ^ y) v x) = (z ^ y) v x. *[para(176(a,1),722(a,1,1))]*.
6279 x v (a ^ (y ^ (x v z))) = x v (a ^ (y ^ z)). *[para(504(a,1),60(a,1,1)),rewrite([5(14),427(13),556(8),277(10),504(20),25(15),4(14),890(14),25(12),4331(11),4586(15)])]*.
6495 ((x v (y ^ a)) ^ z) v (z ^ (y v (a ^ x))) = z ^ (x v y). *[para(2(a,1),67(a,1,1))]*.
6508 (x ^ (y v (z ^ a))) v (u v (x ^ (z v (a ^ y)))) = u v (x ^ (y v z)). *[para(67(a,1),25(a,1,2)),flip(a)]*.
7094 (a ^ x) v (y ^ x) = x ^ ((a ^ x) v y). *[para(128(a,1),4(a,1)),flip(a)]*.
7695 c1 v (c3 v (a ^ c2)) = c1 v c3. *[para(166(a,1),239(a,1,2,2))]*.
7732 (c1 v c3) ^ (c1 v (a ^ c2)) = c1 v (a ^ c2). *[para(7695(a,1),177(a,1,2)),rewrite([2(9)])]*.
9660 x ^ ((x ^ y) v (x ^ z)) = (x ^ y) v (x ^ z). *[para(6(a,1),279(a,1,1))]*.
10028 x v ((x v y) ^ (x v z)) = (x v y) ^ (x v z). *[para(7(a,1),294(a,1,1))]*.
10127 (x ^ (y ^ (z v u))) v ((w ^ (x ^ (y ^ z))) v v5) = (x ^ (y ^ (z v u))) v v5. *[para(298(a,1),233(a,1,2,1,2))]*.
10150 (x ^ (y ^ (z ^ u))) v (w v (y ^ (z ^ (u v v5)))) = w v (y ^ (z ^ (u v v5))). *[para(298(a,1),556(a,1,1,2))]*.
10193 a ^ (c1 v (c2 v (a ^ (c2 v (c3 v x))))) = a ^ (c2 v (c3 v x)). *[para(19(a,1),310(a,1,1)),rewrite([5(10),5089(12),2(9),5(17)])]*.
10202 (x ^ (y ^ z)) v (x ^ (u v y)) = x ^ (u v y). *[para(281(a,1),310(a,1,2,2)),rewrite([281(8)])]*.
10210 (x ^ (y ^ z)) v (z ^ ((x ^ (y ^ z)) v u)) = z ^ ((x ^ (y ^ z)) v u). *[para(224(a,1),310(a,1,1))]*.
10338 x ^ ((y ^ z) v (y ^ x)) = x ^ y. *[para(449(a,1),321(a,1,2)),rewrite([2(5),3(5),9660(4),449(8)])]*.
10367 x v (a ^ (y ^ (z v x))) = x v (a ^ (y ^ z)). *[para(321(a,1),60(a,1,1)),rewrite([5(15),315(14),10150(10),321(20),10127(16),4(11),25(12),4331(11),4586(15)])]*.
10914 x ^ ((y ^ z) v (z ^ x)) = x ^ z. *[para(2(a,1),10338(a,1,2,1))]*.
10915 x ^ ((y ^ z) v (x ^ y)) = x ^ y. *[para(2(a,1),10338(a,1,2,2))]*.
10940 x ^ ((y ^ (z ^ u)) v (u ^ x)) = x ^ u. *[para(224(a,1),10338(a,1,2,1))]*.
11021 x ^ ((y ^ z) v (x ^ z)) = x ^ z. *[para(2(a,1),10914(a,1,2,2))]*.
11183 x ^ ((y ^ z) v (x ^ (u v z))) = x ^ (u v z). *[para(340(a,1),10915(a,1,2,1))]*.
11568 x ^ (y v (x ^ (z v (u v y)))) = x ^ (z v (u v y)). *[para(160(a,1),11021(a,1,2,1))]*.
12001 x v (y v (z v (u v (w ^ x)))) = y v (z v (u v x)). *[para(202(a,1),322(a,1,2,2)),rewrite([391(9)])]*.
12204 x v ((y v x) ^ (x v z)) = (y v x) ^ (x v z). *[para(7(a,1),338(a,1,1))]*.
12396 x v (y v (((x ^ z) v (y ^ u)) ^ w)) = x v y. *[para(357(a,1),280(a,1,2)),rewrite([169(3)]),flip(a)]*.
13792 x v ((y v x) ^ (y v z)) = x v y. *[para(504(a,1),372(a,1,2)),rewrite([25(5),4(4),10028(4),504(8)])]*.
13882 x v ((y v z) ^ (y v x)) = x v y. *[para(2(a,1),13792(a,1,2))]*.
13902 x v (((y ^ z) v x) ^ (u v z)) = x v (y ^ z). *[para(323(a,1),13792(a,1,2,2))]*.
14400 x v ((y v z) ^ (z v x)) = x v z. *[para(4(a,1),13882(a,1,2,1))]*.
14415 x v ((y v z) ^ ((u ^ z) v x)) = x v (u ^ z). *[para(323(a,1),13882(a,1,2,1))]*.
14736 x v ((y v z) ^ (z v (u ^ x))) = x v z. *[para(14400(a,1),149(a,1,2)),rewrite([149(3)]),flip(a)]*.
17603 x v (a ^ (x v y)) = x v (a ^ y). *[para(127(a,1),28(a,1,2)),rewrite([14415(8)]),flip(a)]*.
17613 x v (a ^ (y v x)) = x v (a ^ y). *[para(127(a,1),149(a,1,2)),rewrite([13902(8)]),flip(a)]*.
17730 (a ^ x) v (a ^ y) = a ^ (x v y). *[para(127(a,2),43(a,1,2,2)),rewrite([11183(10),10202(7),2(10),3(10),32(9),124(7),2(11),31(12),5201(12)]),flip(a)]*.
18007 a ^ (c1 v (c2 v (a ^ (c3 v x)))) = a ^ (c2 v (c3 v x)). *[back_rewrite(10193),rewrite([17603(10)])]*.

18151 a ^ (c1 v (c2 v (x ^ a))) = a ^ (c1 v (c2 v x)). [back_rewrite(5089),rewrite([17730(8),5(5)]),flip(a)].
18251 a ^ (c2 v (c3 v (a ^ x))) = a ^ (c1 v (c2 v x)). [back_rewrite(5088),rewrite([18151(16)])].
18578 c1 v (a ^ c3) = c1 v (a ^ c2). [para(17(a,1),17603(a,1,2)),rewrite([17603(7)]),flip(a)].
18580 a ^ (x v (a ^ y)) = a ^ (x v y). [para(17603(a,1),32(a,1,2)),rewrite([3(7),400(6)])].
18588 a ^ (x ^ (y v (a ^ z))) = x ^ (a ^ (y v z)). [para(17603(a,1),166(a,1,2,2)),rewrite([3(8),606(7)])].
18658 a ^ (c1 v (c2 v (c3 v x))) = a ^ (c1 v (c3 v x)). [para(91(a,1),17603(a,2)),rewrite([25(24),4(23),17730(23),25(20),25(19),5(18),25(19),25(18),118(20),18580(21),12001(18),4(17),17730(17),25(14),25(13),4(12),25(12),25(11),25(10),4(9),26(9),25(10),25(9),118(8),118(9)])].
18661 a ^ (c2 v (c3 v x)) = a ^ (c1 v (c2 v x)). [para(98(a,1),17603(a,2)),rewrite([18251(8),18251(15),25(20),4(19),898(19),11568(15),34(13)]),flip(a)].
19033 a ^ (c1 v (c2 v (a ^ x))) = a ^ (c1 v (c2 v x)). [back_rewrite(18251),rewrite([18661(8)])].
19034 a ^ (c1 v (c3 v x)) = a ^ (c1 v (c2 v x)). [back_rewrite(18007),rewrite([19033(10),18658(8),18661(12)])].
19518 c3 v (a ^ (c1 v c2)) = c3 v (a ^ c1). [para(17(a,1),17613(a,1,2))].
19519 c3 v (a ^ c2) = c3 v (a ^ c1). [para(19(a,1),17613(a,1,2)),rewrite([19518(7)]),flip(a)].
19765 a ^ ((a ^ x) v y) = a ^ (x v y). [para(124(a,1),17730(a,1,1)),rewrite([17730(5)]),flip(a)].
22049 (a ^ x) v (x ^ y) = x ^ ((a ^ x) v y). [para(6(a,1),191(a,1,2,1)),rewrite([6(9),2(8)])].
30100 (x ^ y) v ((x v z) ^ (u v y)) = (x v z) ^ (u v y). [para(166(a,1),296(a,1,2)),rewrite([4(5)])].
40090 c2 v ((x v c3) ^ (c3 v (a ^ c1))) = c2 v c3. [para(19519(a,1),14736(a,1,2,2))].
44335 c1 v (a ^ (c3 v x)) = c1 v (a ^ (c2 v x)). [para(19034(a,1),17603(a,1,2)),rewrite([17603(8)]),flip(a)].
46202 (a ^ (x ^ y)) v (y ^ z) = y ^ ((a ^ (x ^ y)) v z). [para(10940(a,1),427(a,2)),rewrite([10210(12),2(9),166(9),2(10)])].
48249 c1 v (a ^ (c2 ^ (x v c3))) = c1 v (a ^ c2). [para(40090(a,1),44335(a,2,2,2)),rewrite([12204(12),18588(11),4(7),17(8),23(9),6279(10),2(6),19(14),17603(15)])].
49221 c1 v (a ^ (c2 ^ c3)) = c1 v (a ^ c2). [para(34(a,1),48249(a,1,2,2,2))].
49233 c1 v (c2 ^ (c3 v (a ^ c1))) = c1 v (c2 ^ c3). [para(49221(a,1),176(a,1,2)),rewrite([25(9),4(8),22049(8),4(7),19519(7)])].
52023 a ^ (x v (y ^ z)) = a ^ ((x v z) ^ (x v y)). [para(537(a,1),19765(a,1,2)),rewrite([4(9),2810(9),17730(8),23(7),23(8),124(7),124(7),10367(11),18580(11)]),flip(a)].
63667 a ^ (x ^ (y v (z ^ u))) = x ^ (a ^ ((y v u) ^ (y v z))). [para(12396(a,1),479(a,1,2,2,2)),rewrite([298(8),2(4),52023(4)]),flip(a)].
63754 x v (y v (z ^ (x v (u ^ z)))) = y v (x v (u ^ z)). [para(921(a,1),390(a,1,2))].
64469 x v (a ^ ((y v x) ^ z)) = x v (a ^ (y ^ z)). [para(745(a,1),941(a,1,1,2)),rewrite([52023(5),120(3),2(10),63667(11),120(8),293(11),4(10),6279(10),25(10),4(9),17730(9),30100(6),6279(6),2(10),63667(11),120(8),293(11),4(10),6279(10)])].
69039 x v ((x v y) ^ (z v (a ^ x))) = x v ((x v y) ^ z). [para(1108(a,1),7094(a,1,2)),rewrite([52023(5),118(4),25(9),46202(8),4(7),64469(7),7(4),52023(14),118(13),4(16),148(16),2(11),1108(11)])].
104056 c1 v (c2 ^ (c1 v c3)) = (c1 v c2) ^ (c1 v c3). [para(7732(a,1),6495(a,1,2)),rewrite([2(4),2(9),4(15),5(15),267(14),25(13),4(12),19519(12),176(13),4(9),2(10),69039(11),2(6),4(13),2(14)])].
104095 c1 v (c2 ^ c3) = (c1 v c2) ^ (c1 v c3). [para(49233(a,1),6508(a,1,2)),rewrite([2(5),18578(6),25(13),4(12),63754(13),25(9),4(8),22049(8),4(7),19519(7),49233(9),104056(12)])].
104096 $F. [resolve(104095,a,21,a)].

============================== **end of proof** ==========================

============================== **STATISTICS** ============================

```
Given=4826. Generated=39046099. Kept=104089. proofs=1.
Usable=4530. Sos=19712. Demods=22873. Limbo=11, Disabled=79850. Hints=0.
Kept_by_rule=0, Deleted_by_rule=26712.
Forward_subsumed=19936868. Back_subsumed=810.
Sos_limit_deleted=18978430. Sos_displaced=63145. Sos_removed=0.
New_demodulators=92229 (6 lex), Back_demodulated=15878. Back_unit_deleted=0.
Demod_attempts=1136610097. Demod_rewrites=133339635.
Res_instance_prunes=0. Para_instance_prunes=0. Basic_paramod_prunes=0.
Nonunit_fsub_feature_tests=0. Nonunit_bsub_feature_tests=0.
Megabytes=136.64.
User_CPU=963.34, System_CPU=15.07, Wall_clock=979.
```

============================== **end of statistics** =====================

============================== **end of search** =========================

**THEOREM PROVED**